Action4D: Online Action Recognition in the Crowd and Clutter

Quanzeng You Hao Jiang
Microsoft Cloud & AI
One Microsoft Way, Redmond, WA 98052
{quyou, jiang.hao}@microsoft.com

Abstract

Recognizing every person's action in a crowded and cluttered environment is a challenging task in computer vision. We propose to tackle this challenging problem using a holistic 4D "scan" of a cluttered scene to include every detail about the people and environment. This leads to a new problem, i.e., recognizing multiple people's actions in the cluttered 4D representation. At the first step, we propose a new method to track people in 4D, which can reliably detect and follow each person in real time. Then, we build a new deep neural network, the Action4DNet, to recognize the action of each tracked person. Such a model gives reliable and accurate results in the real-world settings. We also design an adaptive 3D convolution layer and a novel discriminative temporal feature learning objective to further improve the performance of our model. Our method is invariant to camera view angles, resistant to clutter and able to handle crowd. The experimental results show that the proposed method is fast, reliable and accurate. Our method paves the way to action recognition in the real-world applications and is ready to be deployed to enable smart homes, smart factories and smart stores.

1. Introduction

Action recognition is a key task in computer vision. Even though human vision is good at recognizing subtle actions, computer vision algorithm still cannot achieve the same robustness and accuracy. The difficulty is largely caused by the variations of the visual inputs. The input video may be crowded and cluttered. People may have different clothing, different body shapes and are highly articulated. They may perform the same action in slightly different ways. The camera viewing angles can be drastically different so that the same action in the training videos may look quite different from those in the testing videos.

To tackle the above challenges, in this paper, we propose a novel 4D method for robust action recognition. The input of our method is a 4D volume of the dynamic environment

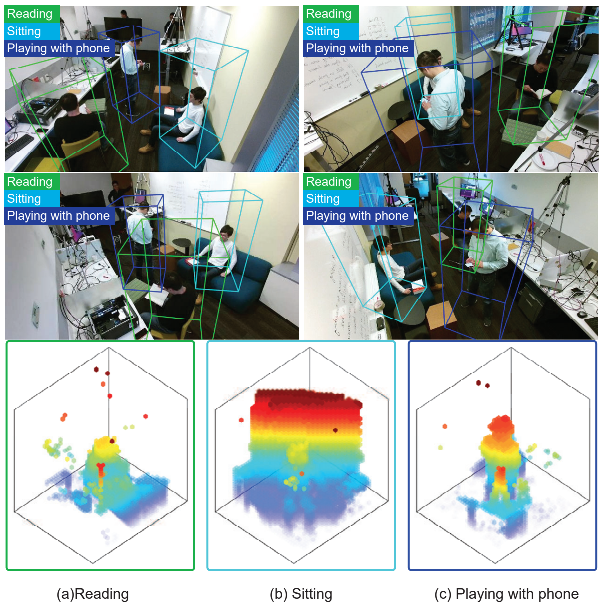


Figure 1. Examples of the inference results of our action recognition system on a crowded and cluttered environment with four Kinect V2 cameras. The top two rows show the projected 3D bounding boxes on each camera view given by our multiple-people tracker. The last row shows the volumes generated from the four calibrated depth cameras for each subject with different actions.

constructed from multiple calibrated RGBD cameras. Figure 1 illustrates our scheme. The proposed method tracks each individual person using the 4D representation and recognizes their actions. It is view invariant, able to handle crowd and clutter, and scalable to applications in a huge space with hundreds of cameras.

Recognizing multiple people's actions in cluttered 4D volume is a new and challenging problem. To the best of our knowledge, our method gives the first solution to this problem. In particular, we propose a novel Action4DNet to recognize the action of each subject in a cluttered environment using online 4D modeling. Our work has the following contributions:

- We tackle the new problem of recognizing multiple people's actions in cluttered 4D volume data.

- We propose a new people detection and tracking method using the 4D volume data in real time.

- We propose a new deep neural network, Action4DNet, for action recognition. We design an adaptive convolutional layer to deal with the noise introduced from multiple camera sensors. We also propose a new discriminative loss for better temporal feature learning in sequential action recognition. To the best of our knowledge, our approach is the first attempt to apply deep neural networks to cluttered "holistic" 4D volume data for online frame-wise action recognition.

- We collect and label a new 4D dataset in our experimentation. There is no existing 4D action recognition dataset that includes multiple people and clutter. We will publish the dataset.

- Our proposed method is resistant to crowd and clutter, and it can be directly used in complex real-world applications.

1.1. Related works

In previous studies, most action recognition methods work on single view 2D videos. Accumulated foreground shape [1] has been used to recognize the actions in the KidsRoom project. In [28], shape context is used to model the whole body configuration in action recognition. Apart from RGB color, motion is also a useful feature for action recognition [9]. Other popular handcrafted features for action recognition include spatial-temporal features [14] and spatial-temporal volumes [4]. Based on these features, action detection and recognition can be formulated as a matching problem. By careful design, we do not even need to directly extract the features; the space and time matching can be efficiently solved using low rank analysis [23].

In recent years, deep learning has been widely used in action recognition and detection using RGB videos [12, 30, 24, 8, 7]. These deep learning methods use multiple streams such as color, motion, body part heat map and find actions in the spatial-temporal 3D volume. Single view depth images have also been used in action recognition [26]. However, training classifiers for action recognition using 2D RGB or depth videos is a challenging task. It requires training data to include all kinds of variations about camera settings, people clothing, object appearances, and backgrounds.

Most of the current 3D action recognition methods depend on Kinect 3D skeleton extraction [17, 21, 22], which can relieve the view dependency issue in 2D action recognition. Unfortunately, Kinect skeleton estimation becomes unreliable in cluttered environments. In addition, 3D skeletons alone are insufficient for action recognition. For instance, disambiguating actions such as playing with phone and reading a book is tricky without knowing the objects in people's hands. 3D people volume from visual hull [15] has also been extensively used in action recognition [5, 29, 11]. Traditional visual hull methods usually need special blue/green or static background and background subtraction to single out people from the background. This greatly limits its usability in real-world applications.

In contrast, our method works directly on cluttered 4D volume data. The volume representation includes not only people but also the objects they are interacting with. Without the dependency on people segmentation, our method can be robustly applied to action recognition in crowded and cluttered environments.

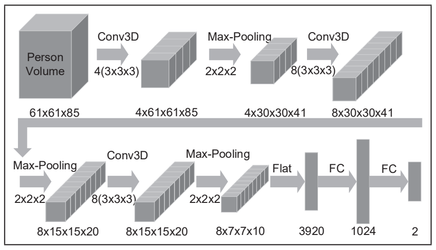
2. Method

Our task is to recognize individuals' actions in a cluttered and crowded environment. Our method starts with the construction of 3D volume representation of the whole scene at each time instant. Then, we propose a new people detection and tracking method using sequential 3D volume data of the whole scene. In such a way, we can crop each person-centered 3D volume at each time instant. These associated 3D volume sequences by our 4D tracker are used as input to build our Action4DNet. The details are discussed in following sections.

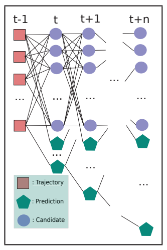
2.1. People detection and tracking

Detecting each subject in the scene is a necessary step before we can recognize the action of each individual. For action recognition, we also need to observe every subject in a duration. We thus need to track each person in the scene. Tracking also helps remove false people detections and recover the missing ones. Most of the previous multiple people tracking methods usually use background subtraction to remove the background clutter. Unfortunately, background subtraction or figure/ground separation is hard for unconstrained dynamic environment. Our 4D tracker does not need figure/ground separation and is able to work on the noisy 4D data directly.

Given a set of calibrated RGBD images, we build the 3D point cloud of the whole scene. The volumes are built on top of the 3D point cloud. We set the occupancy of a voxel $\boldsymbol{O}(i)$ to one if there is a point in it. These voxels are on the scene surface of the environment. It is also possible to fill the internal voxels of each object. However, our experiments suggest that action recognition does not benefit much from such a denser representation. We thus only use the surface volumes in this work.

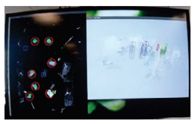


(a)



(b)

Figure 2. (a): People classification CNN. (b): We find disjoint paths on the tracking graph.



(a) Tracking results

(b) Two side-view RGB images of our lab

Figure 3. Sample tracking results of the proposed method. (a) Visualization of our tracking results. On the left of the screen is the top-down view with each numbered red circle representing one person. On the right is the real-time 3D point cloud. (b) We use two other RGB cameras to capture the corresponding side views of our lab.

2.1.1 People candidate proposal

We use a light-weight people candidate proposal scheme as follows: let $f(x, y, z)$ be the volume data and assume $z = 0$ is the ground plane. The top-down envelop image is $g(x, y) = \max_z(z\mathbb{1}(f(x, y, z)))$, where $\mathbb{1}(\xi)$ is an indicator function which equals 1 if $\xi > 0$, otherwise 0. Based on the observation that each potential object corresponds to at least one local maximum on g, we use a simple Gaussian filter to extract the candidates. The local maxima are found on the Gaussian filtered top-down envelope using non-maximum suppression. Each candidate volume is a cuboid around a local maximum with a given width and height. Currently, we set the height of the cropped volumes to be the height of the whole scene volume.

We train a 3D CNN to classify each candidate volume to be people or non-people. The CNN structure of our people classifier is shown in Figure 2 (a), which consists of a sequence of 3D convolution layers, ReLUs and pooling layers (ReLUs are not shown), followed by a multi-layer perceptron (MLP). The 3D people classifier gives the probability of each candidate 3D bounding box containing a person. Even with just a few thousand frames of training data, the people detector can achieve high accuracy to support the following data association for people tracking.

2.1.2 Data association

With the extracted candidates, people tracking can be formulated as a path following problem. We try to link the detected trajectories to the detections in the current frame t and the next n frames. Here n is a small number, *e.g.*, three.

The tracking graph is shown in Figure 2 (b). There are three kinds of nodes in the graph: the rectangle nodes represent the trajectories already formed, the oval nodes represent candidates, and the pentagon nodes are the prediction nodes. The number of prediction nodes equals the number of candidate nodes plus the number of the prediction nodes at previous time instant. The edges indicate possible matches between nodes. The edge weights are determined by the difference of the probabilities from our 3D people classifier, the Euclidean distance, the occupancy volume difference, and the color histogram difference between neighboring nodes. The trajectory node also has a weight inversely proportional to the trajectory length. To track objects in the scene, we find the extension of each trajectory from time $t - 1$ to $t + n$, so that these paths pass each trajectory node and all the paths are node disjoint.

This optimization problem can be reduced to a min-cost flow problem and it can be solved efficiently using a polynomial algorithm [19]. Each trajectory is only extended to the neighboring nodes within a radius d_L, which is determined by the max speed of a person and the frame rate of the tracking algorithm.

After the optimization, we extend each existing trajectory by one-unit length. We remove trajectories with low people score, which is computed as the weighted sum of the current people probability and the previous people score. And, we include new trajectory for each candidate node at time t that is not on any path. The new set of trajectories are used to form a new graph for the next time instant.

Our people detection and tracking algorithm is robust against clutter and crowd. Figure 3 shows sample results from our 4D tracking over a few thousand frames. The tracker is able to handle cases such as putting a box above the head as shown in Figure 3 (b).

2.2. Action recognition

The above tracker gives us accurate 3D location of each subject at each time instant, which can be used to crop out 3D volumes for action recognition. Figure 4 shows the cropped volume representations, where persons are at the center. Even with the cluttered background, the volume representation clearly shows the action of a person. As a matter of fact, the background objects are desirable for action recognition because of their context information.

We process the 4D volume (sequence of 3D volumes) data to infer the action at each time instant. There are many other clues that can be used to infer the action of a person, *e.g.*, the body poses, the movement of body parts, and the objects the subject is handling. For instance, if we see a chair underneath a person, we can infer that the person is

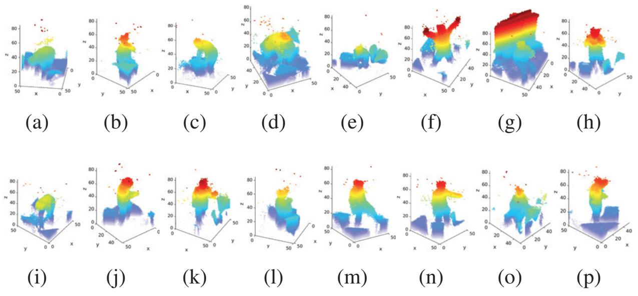


Figure 4. We recognize actions using volumes (color represents the height of each voxel). The action can be easily identified even from a static snapshot. The actions are (a) bending, (b) drinking, (c) lifting, (d) pushing/pulling, (e) squatting, (f) yawning, (g) calling, (h) eating, (i) opening drawer, (j) reading, (k) waving, (l) clapping, (m) kicking, (n) pointing, (o) sitting, and (p) browsing cell phone. These real-time generated volumes are fed into our Action4DNet for action recognition.

sitting. Potentially, the position or the speed of each person can also be used to infer specific actions. However, in this paper we depend on the volume data alone for building our 4D action recognition model.

We construct deep convolutional neural network, Action4DNet, for accurate action recognition. The input 4D volumes go through a sequence of 3D convolution layers combined with 3D pooling layers to produce action features. Meanwhile, we also propose to use an auxiliary attention net, which will be discussed in more details in the following subsections. These features at each time instant are fed into a Recurrent Neural Network (RNN) to aggregate the temporal information for final action classification.

In the following, we present the network structures in more details.

2.2.1 Attention Action4DNet

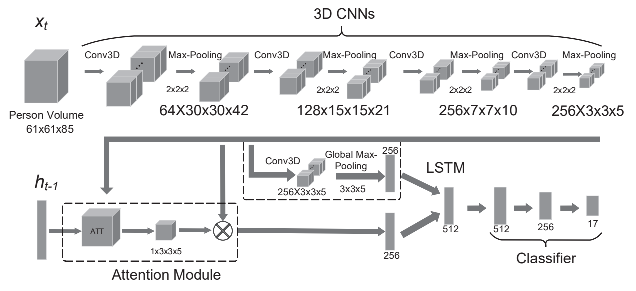


Figure 5. Our proposed attention Action4DNet.

Figure 5 shows the proposed neural network architecture for action recognition using the volumetric data. It starts with several 3D convolution layers followed by 3D max-pooling layers. Then, attention model [2, 18] is employed to automatically learn the most relevant local sub-volume features, and global max-pooling [16] is used for global features. Both features are the inputs to the Recurrent Neural Network (we use LSTM) for action classification.

Let $\boldsymbol{V} \in \mathbb{R}^{F\times L\times W\times H}$ be the output from the last 3D convolution layer, where F is the number of filters, L, W and H are the size of the 3D output. In particular, each location in the 3D output can be represented as $\boldsymbol{v}_{ijk} \in \mathbb{R}^F$ for $1 \le i \le L$, $1 \le j \le W$ and $1 \le k \le H$. The attention weights for all $\boldsymbol{v}_{ijk}$ are computed as

$$\boldsymbol{\beta}_{ijk} = \boldsymbol{h}_{t-1}^T \boldsymbol{U} \boldsymbol{v}_{ijk} \tag{1}$$

$$\boldsymbol{\alpha} = \texttt{softmax}(\boldsymbol{\beta}), \tag{2}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{L\times W\times H}$ is the attention weights, $\boldsymbol{U} \in \mathbb{R}^{D\times F}$ is the weight matrix to be learned and $\boldsymbol{h}_{t-1} \in \mathbb{R}^D$ is the previous hidden state of size D from the Recurrent Neural Network. In such a way, the network is expected to automatically discover the relevance of different sub-volumes for different actions.

Next, the local feature $\boldsymbol{v}$ is computed as the weighted sum of all the sub-volume features $\boldsymbol{v}_{ijk}$

$$\boldsymbol{v} = \sum_{i,j,k} \boldsymbol{\alpha}_{ijk} \boldsymbol{v}_{ijk}. \tag{3}$$

In addition, we employ a 3D convolution layer followed by a global pooling layer to obtain the global feature $\boldsymbol{g}$ (see Figure 5). Next, both the global feature $\boldsymbol{g}$ and the local attention feature $\boldsymbol{v}$ are supplied to the LSTM cell to capture the temporal dependencies. The action classification model, which is a multi-layer perceptron (MLP), takes the hidden state from the LSTM cell as input to recognize different actions at each time instant.

2.2.2 Adaptive convolutional layer

In previous section, we describe our attention Action4DNet using standard 3D convolutional neural networks. In real-world environments, camera sensors and calibration errors can introduce noises to the generated volumes. In our volume representations, these noises could lead to different activation outputs for a regular convolutional layer. It is attractive if the model itself can adapt to the noise. We propose the adaptive convolutional layer, which is designed with an extra adaptive activation mechanism.

Again, let $\boldsymbol{V} \in \mathbb{R}^{F\times L\times W\times H}$ be the output from the last 3D convolution layer. Then, we attach another convolution layer with two $1 \times 1 \times 1$ kernels followed by a softmax operator. The output is denoted as $\boldsymbol{Z} \in \mathbb{R}^{2\times L\times W\times H}$, which serves as the adaptive probability for each location in V. We produce the new outputs at location (i, j, k) as

$$\boldsymbol{V}'[:, i, j, k] = \boldsymbol{Z}_{1ijk} * \boldsymbol{V}[:, i, j, k], \tag{4}$$

where $*$ is the product between a scalar $\boldsymbol{Z}_{1ijk}$ and a vector $\boldsymbol{V}[:,i,j,k] \in \mathbb{R}^F$. This layer can be inserted into any regular 3D convolutional layers. We call them adaptive convolutional layers.

2.2.3 Discriminative temporal feature learning

RNNs are designed for capturing temporal dependencies. State-of-the-art action recognition models also apply RNNs to action recognition [25] in order to understand and incorporate temporal information in videos. However, the temporal transitions are more difficult to capture in continuous domains, such as videos than in discrete domains, such as natural languages. Recently, *optical flow* has been widely employed to assist the model for better temporal feature learning. The computation of optical flow requires more computing power. In addition, it is more difficult in our 4D scenario, due to the noise in our large volume data.

Instead, we expect the model can learn to distinguish the stepwise states by only looking at the sequential data. To achieve this goal, we propose a margin-ranking loss, which tries to differentiate the temporal features in a given training sequence. Let $\boldsymbol{H} = \{\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_n}\}$ be the hidden states from the recurrent neural network (see Figure 5). We add additional semantic layer with weight $\boldsymbol{W}$ to map $\boldsymbol{h'_i} = \boldsymbol{W}\boldsymbol{h_i}$ and define the loss as

$$L(\boldsymbol{H'}) = \sum_{i=1}^{n-2} \max(0, c + \texttt{score}(\boldsymbol{h'_i}, \boldsymbol{h'_n}) - \texttt{score}(\boldsymbol{h'_{n-1}}, \boldsymbol{h'_n})) \tag{5}$$

where $\boldsymbol{H'} = \{\boldsymbol{h'_1}, \boldsymbol{h'_2}, \ldots, \boldsymbol{h'_n}\}$ is the set of the mapped hidden states, c is a constant and $\texttt{score}(\cdot,\cdot)$ computes the similarity between its two inputs. In this work, we adopt the cosine similarity function.

The above loss attempts to guarantee that within any given training sequence, the last frame has a larger similarity with the second to the last frame than all other previous frames. In general, this constraint is true in videos as well as in our 4D case. In particular, for training sequences where all the frames have the same label, this loss function differentiates the states at different input frames. At the same time, the cross-entropy loss for action recognition classifier tries to maintain correct prediction on these differentiated states. Our experiments suggest that this mechanism leads to better action recognition performance for our Action4DNet model.

3. Experimental results

In this section, we evaluate the proposed 4D approach for action recognition and compare our approach against different competing methods.

3.1. Ground truth experimentation setup

To evaluate the performance of our method, we collect a 4D action recognition dataset. We set up three different environments (Env1, Env2 and Evn3) with different number of Kinect V2 cameras to capture the RGBD images and then we generate 4D volume representation of the dynamic scene. The three environments are located at different rooms with different backgrounds. We label the videos in a per-frame fashion: each video frame has an action label. We also evaluate all action recognition models using the per-frame accuracy. The statistics of our dataset are summarized in Table 1.

Envs	# of cameras	# of subjects	# of volumes
Env1	4	15	90K
Env2	8	12	64K
Env3	7	9	34K

Table 1. Our dataset from three different environments.

The scene includes not only people but also objects such as sofa, tables, chairs, boxes, drawers, cups, and books. There are over 20 different subjects in the dataset. They have different body shapes, gender and heights. The dataset includes 16 actions in the everyday life: drinking, clapping, reading book, calling, playing with phone, bending, squatting, waving hands, sitting, pointing, lifting, opening drawer, pull/pushing, eating, yawning, and kicking. Each action can be done in a standing or a sitting pose. Here, action "sitting" means sitting without doing anything.

We compare our proposed method against different baseline methods. The baselines include:

- **ShapeContext256 and ShapeContext512**: 3D Shape context is a 3D version of the shape context [3] descriptor. The 3D shape context has the height axis and the angle axis uniformly partitioned, and the radial axis logarithmically partitioned. We test two versions of the 3D shape context: **ShapeContext256** has 256 bins and **ShapeContext512** has 512 bins. We build a deep network whose input is the 3D shape context descriptors. The network uses an LSTM network to aggregate the temporal information.

- **Moment**: Moment is another popular shape descriptor. We use the raw moments up to order 4. Similar to the above shape context approach, the moment descriptor is fed into a CNN for action recognition.

- **Skeletons**: OpenPose [6] is one of the state-of-the-art stick figure detectors on RGB images. We normalize the positions of the joints of each subject using the neck point and then concatenate the xy coordinates into a feature vector. We train a deep network using similar approach to the above shape context method.

- **Color+Depth**: In this method, we find the bounding boxes of each person based on our tracking result. We crop the color and depth images of each person in the video from all the cameras. We train a deep neural network using the cropped color and depth images and their action labels. To be fair, we do not use motion in all the methods in this paper.

- **PointNet**: PointNet [20] is one of state-of-the-art deep learning methods for object recognition and semantic segmentation on 3D point clouds. We extend the PointNet model to include an LSTM layer so that it can handle sequential data for action recognition. The network can be trained end-to-end using the point clouds from multiple RGBD images.

- **I3D and NL-I3D**: Inflated 3D ConvNet [7] (I3D) achieves the state-of-the-art action recognition on RGB videos. We also compare with non-local I3D [27] (NL-I3D), which introduces non-local operations for better long-range dependencies modeling.

- **SparseConvNet** SparseConvNet [10] defines submanifold convolution, which keeps track of "active" site to reduce computational overhead. We train SparseConvNet using the 3D volumes along with an LSTM head to recognize actions in 3D streams.

All the models are implemented using PyTorch. For I3D and NL-I3D, we use the pre-trained models on Kinetics dataset [13] and fine-tune them on our dataset. All other baselines and our proposed models are trained from scratch. We use the same training, testing and validating splits when evaluating different methods. To make the models agnostic to the number of cameras, we train **Skeleton**, **I3D**, **NL-I3D** on single camera frames. During testing, we aggregate the predictions from different cameras to obtain the final action recognition results. **Color+Depth** also depends on the number of cameras. However, when trained on single camera frames, its performance is much worse. We only report its performance on test one and two (using all four camera frames for training and testing) under environment one.

In our experimentation, we extract the 4D volume representation for each person based on our 4D people tracker. Given each person's location, we extract a volume centered at that location. This volume is set large enough to cover a person with different poses. In particular, we experiment with different voxel sizes. Table 2 shows the results of two models. **Conv3D + ATT** is the model in Figure 5 and **Conv3D** is with similar architecture without the attention branch. The proposed adaptive convolution layer and the discriminative loss have not been applied in this experiment. The results in Table 2 shows that we can achieve better performance with smaller voxel size. In the following experiments, we will use the 25mm voxels to evaluate the proposed models.

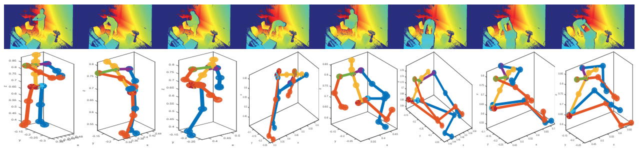


Figure 6. Kinect V2 skeleton estimation is prone to errors if there is clutter in the scene. Row one: Depth images. Row two: Normalized skeletons from Kinect V2.

Model	Voxel Size	Volume Size	Acc
Conv3D	50mm	$31 \times 31 \times 43$	71.1
Conv3D + ATT	50mm	$31 \times 31 \times 43$	74.8
Conv3D	25mm	$63 \times 63 \times 85$	80.5
Conv3D + ATT	25mm	$63 \times 63 \times 85$	82.5

Table 2. Accuracies (Acc) of two Conv3D models on similar volume coverages with different voxel sizes on the data of Env1.

Apart from the target subject, background clutter and other subjects in the scene are also included in the cropped volume as shown in Figure 4. Potential approaches, such as semantic segmentation and 3D skeleton estimation, can be used to separate a person from the clutter. However, the results can be unreliable in a "cluttered" environment.

For instance, Figure 6 shows that the skeleton estimation from Kinect V2 becomes increasingly unreliable as the background clutter increases. When people interact with large objects and their body parts are occluded by these objects, the skeleton estimation fails.

In this paper, we therefore do not depend on the semantic segmentation and 3D skeleton estimation. Instead, we use the full 4D volume data that contains every bit of information for action recognition. In the following, we show the experimental results on the ground truth data.

3.2. Ground truth experimentation

We conduct the following three tests on our dataset. (1) **Test one**: we use 14 single-subject videos performed by different subjects from **Env1**. The training is on ten videos and testing is on three videos. One video is used for validation. It has a total of 68K frames in the training videos, 6K frames in the validating videos and 10K in the testing videos. (2) **Test two**: we take the trained models from test one and evaluate them on 4 multiple-subject videos also collected from **Env1**, which include 3, 3, 3, and 2 people respectively. It has a total of 6K frames for all the multiple-subject testing videos. (3) **Test three**: we also conduct a **cross-environment test** to further study the robustness of different approaches. We train all models on data from **Env1** and **Env2**. We test all models on data collected from **Env3**, which has some non-overlap subjects with Env1 and Env2 as well as single and multiple person videos. One video from **Env3** is used as validating video for model selection.

We report two accuracy numbers for each test. Acc is stricter: we deem action recognition is correct if and only if the action prediction result matches the ground truth label of the corresponding video frame. One issue about this criterion is that at the action boundaries, accurate labeling is hard. For transient actions, the small offset of labeling may cause the mismatch between detection results and the ground truth. To remedy this issue, we define another accuracy, revised accuracy (RAcc). For RAcc, an action classification is correct if and only if the predicted action label is the same as the ground truth label of a frame within the window of plus/minus three relative to the current video frame.

Models	Person 1		Person 2		Person 3		Average	
	Acc	RAcc	Acc	RAcc	Acc	RAcc	Acc	RAcc
ShpCtx256	56.9	63.1	51.5	56.5	55.7	61.6	54.7	60.5
ShpCtx512	55.2	60.5	47.6	53.1	56.8	62.6	53.4	58.9
Moments	37.4	44.9	44.9	54	38.4	47.1	40.1	48.6
Color+Depth	53.6	60.5	64.1	71.7	52.7	60.1	56.6	63.9
Skeleton	66.7	72.2	72.1	79.1	56.8	62.8	64.9	71.0
PointNet	58.9	63.7	76.5	79.1	57.9	63.8	58.7	63.5
SparseNet	69.5	76.1	71.9	79.9	69.7	76.4	70.3	77.4
I3D	77.5	84.6	78.8	87.8	74.2	82.4	76.7	84.8
NL- I3D	73.7	81.3	78.8	88.1	74.2	82.9	75.5	84.0
Action4DNet (w/o ATT)	83.6	**90.2**	79.2	86.8	79.1	85.9	80.6	87.5
Action4DNet	**84.0**	89.6	**84.7**	**91.9**	**83.6**	**90.0**	**84.1**	**90.4**

Table 3. Evaluation of the proposed models and several baselines on test one. We show percentages of both the accuracies (Acc) and the revised accuracies (RAcc) of all the evaluated models.

Table 3 shows the accuracies of different competing methods in ground truth test one. In this test, our proposed Action4DNet trained with adaptive convolutional layer and the proposed discriminative loss achieves the highest average revised accuracy (RAcc) 90.0%. We also train the model without using the attention model, which obtains worse performance than Action4DNet. However, it still performs better than all baselines. Our method's accuracy improves by more than 30% over the competing methods such as ShapeContext, Moment, Color+Depth, Skeleton and PointNet and by about 6% over recent deep learning based approaches on RGB videos including I3D, NL-I3D and SparseNet. We also achieve the highest accuracies in each individual test.

These results are not a surprise. The handcrafted features such as shape context and moments are not as strong as the learned features from deep learning especially when there is strong background clutter. The PointNet gives low accuracies in this experiment. This is likely due to the strong clutter and because PointNet has to sample the point clouds to fit into the GPU memory. The Color+Depth and Skeleton approaches perform better than other handcrafted feature methods, but they give much worse results than our proposed method. I3D and NL-I3D show better performance over other approaches. However, both methods are also dependent on the camera views: if the camera settings are different, we have to retrain the model. In contrast, our proposed method can be used in different camera settings without retraining. The input to SparseNet is view-independent as well. However, it shows worse performance than our model. The following test two and test three confirms the generability of our approach.

Models	Acc	RAcc
ShapeContxt	37.5	43.6
ShapeContxt16	34.2	39.1
Moments	36.2	44.5
Color+Depth	46.1	56.6
Skeleton	53.8	62.0
PointNet	58.9	64.6
SparseNet	60.4	68.3
I3D	58.1	66.7
NL-I3D	56.4	64.2
Action4DNet (w/o ATT)	79.9	87.1
Action4DNet	**86.3**	**93.3**

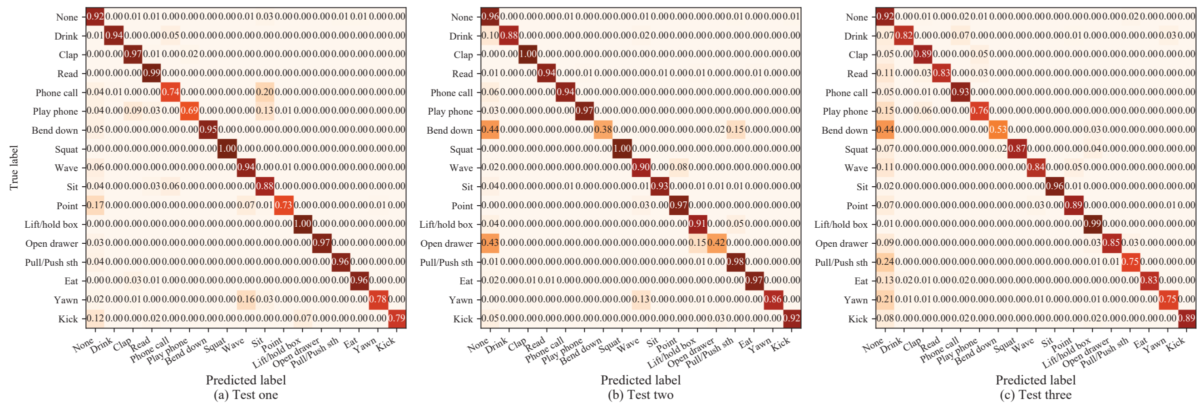
Table 4. Evaluation of the proposed models and several baselines in ground truth test two, which involves multiple people.

We use the same model trained in **test one** to evaluate all the multiple people videos in **test two**. As shown in Table 4, our method still achieves the highest accuracy among all the methods. PointNet and SparseNet show less performance degradations due to their color agnostic inputs. All other competing approaches show worse accuracies due to multiple people mutual occlusions and background clutter. As a matter of fact, our Action4DNet even shows better performance than in **test one**. This could be attributed to the short duration and thus less variation of actions in each of the testing video in **test two**.

Models	Acc	RAcc
Skeleton	45.0	49.4
PointNet	49.4	53.8
SparseNet	68.2	73.3
I3D	58.1	65.3
NL-I3D	61.3	68.3
Action4DNet (w/o ATT)	74.8	80.5
Action4DNet	**81.4**	**87.0**

Table 5. Results on cross-environment testing. We train all the models on data from Env1 and Env2 and test them on Env3.

Due to space limit, we only include the performance of deep learning approaches for **test three**. The results are shown in Table 5. Again, the proposed model shows bet-



(a) Test one (b) Test two (c) Test three

Figure 7. Confusion matrix for ground truth test of our Action4DNet model.

ter performance against all baselines. In addition, the accuracies of our model are comparable with our results in Table 3, which is tested in the same environment. This suggests that our approach is general and robust against background changes. Other approaches, including I3D and NL-I3D are severely impacted by the different backgrounds and lightings in different environments.

Table 3, Table 4 and Table 5 show that our proposed method consistently gives much better results than all the competing methods. The high accuracy also benefits from our reliable 4D people tracker, which obtains 100% tracking rate for all the testing and training videos. Our method is also fast, with a single GTX1080 TI, our method is able to track 10 people and infer their actions at 15 frames per second (FPS) on volumes with $50mm \times 50mm \times 50mm$ voxel size. On the $25mm \times 25mm \times 25mm$ voxels, it is possible to recognize actions at 25 FPS on a single person.

Figure 7 shows the confusion matrices of our Action4DNet on the three different tests. It is interesting to see that there are many missing detections in **test two** and **test three**. Especially, for the *bend down* action, both test two and test three have over 40% missing recognitions. This is potentially due to the large variations of this action and the inconsistent labeling standards used by different ground truth labelers. Meanwhile, our method also confuses some actions as seen in Figure 7. This is mostly due to the noisy data from the Kinect sensor. Using better depth cameras and better time synchronization, our action recognition results can be further improved. Moreover, we can further include other voxel attributes such as color and use multi-resolution volume data to achieve more robust results.

3.3. Ablation study

We evaluate the impact of the proposed adaptive convolution layer and the discriminative temporal feature learning on action recognition performance. Table 6 lists the results. We show the accuracies of Action4DNet on **test one**, **test two** and **test three** respectively. The proposed adaptive convolution layer introduces more parameters. Thus, it may not help the model without the discriminative temporal feature learning as shown in **test one**. However, the results in **test two** and **test three** suggest that the adaptive convolution layer indeed improves the generability of the model on different settings. The discriminative loss improves the performance over baseline Action4DNet-A-D in all three tests. Overall, the results indicate that the two proposed mechanisms are effective in learning better action recognition models from 4D volumes.

Models	Test one	Test two	Test three
Action4DNet-A-D	82.5	82.5	76.4
Action4DNet-D	81.6	85.4	80.6
Action4DNet-A	82.6	85.3	81.3
Action4DNet	84.1	86.3	81.4

Table 6. Ablation study on the proposed adaptive convolution and the discriminative loss. We use "-A" and "-D" to represent the model variations without adaptive convolution and without discriminative loss respectively.

4. Conclusion

We propose a novel online 4D action recognition method, the Action4DNet, which is able to generate 4D volumes of the environment, track each person in the volume and infer the actions of each subject. Our method is able to handle multiple people and strong clutter. In particular, the proposed adaptive convolution layer and the discriminative temporal feature learning objective further improve the performance of our model. Our experimental results under different settings confirm that our method gives better performance over different competing methods. The proposed method can be deployed to enable different applications to enhance how people interact with the environment.

References

[1] Kidsromm. `http://vismod.media.mit.edu/vismod/demos/kidsroom/kidsroom.html`. 2

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 4

[3] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 24(4):509–522, 2002. 5

[4] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, volume 2, pages 1395–1402 Vol. 2, 2005. 2

[5] Cristian Canton-Ferrer, Josep R Casas, and Montse Pardas. Human model and motion based 3d action recognition in multiple view scenarios. In *2006 14th European Signal Processing Conference*, pages 1–5. IEEE, 2006. 2

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017. 5

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017. 2, 6

[8] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[9] Alexei A Efros, Alexander C Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *ICCV 2003*, page 726. IEEE, 2003. 2

[10] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *Proceedings of the IEEE Computer Vision and Pattern Recognition CVPR, Salt Lake City, UT, USA*, pages 18–22, 2018. 6

[11] Michael B. Holte, Cuong Tran, Mohan M. Trivedi, and Thomas B. Moeslund. Human action recognition using multiple views: A comparative perspective on recent developments. In *Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding*, pages 47–52, 2011. 2

[12] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *IEEE International Conference on Computer Vision, ICCV 2017*, pages 4415–4423, 2017. 2

[13] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 6

[14] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 432–, 2003. 2

[15] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. 2

[16] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 4

[17] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1647–1656, 2017. 2

[18] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014. 4

[19] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998. 3

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 6

[21] Hossein Rahmani and Mohammed Bennamoun. Learning action recognition model from depth and skeleton videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5832–5841, 2017. 2

[22] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[23] E. Shechtman and M. Irani. Space-time behavior based correlation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 405–412 vol. 1, 2005. 2

[24] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip H. S. Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[25] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

[26] Pichao Wang, Wanqing Li, Zhimin Gao, Yuyao Zhang, Chang Tang, and Philip Ogunbona. Scene flow to action map: A new representation for rgb-d based action recognition with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[27] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 6

[28] Yang Wang, Hao Jiang, Mark S Drew, Ze-Nian Li, and Greg Mori. Unsupervised discovery of action classes. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1654–1661. IEEE, 2006. 2

[29] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer vision and image understanding*, 104(2-3):249–257, 2006. 2

[30] Mohammadreza Zolfaghari, Gabriel L. Oliveira, Nima Sedaghat, and Thomas Brox. Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2