Human Gesture Recognition for Drone Control

Heba Aly [1] **Carolin Arnold** [1] **Ananth Hari** [1] **Phil Nguyen** [1] **Snehesh Shrestha** [1]

Abstract

We propose a Maximum Entropy Markov Model (MEMM)-based system to recognize nine gestures relevant to controlling drones. The MEMM graph captures the gestures structure and variation over time. In addition, for inference, we employ an on-the-fly graph built to capture the gestures rate variations. We evaluated our model on a public data-set with 21 people performing the gestures in different environments. The proposed model achieves an accuracy of 80% which gives a relative improvement of 21% over our HMM-based baseline and up to 8.1% over other comparison models.

1. Introduction

Although drones have become very popular, the traditional way of controlling them (with a radio controller) is difficult due to the drones' six degrees of freedom (6DoF), which substantially increases the controller's complexity. It would be a lot more natural to be able to use gestures to control the drones instead. There already exists a standard set of gestures that is used by airtraffic controllers to signal an aeroplane and helicopter pilots (FAA, 2017). The advantage of these gestures are they are standardized and the movements are dramatic allowing air vehicles to be able to see from a distance. Therefore, we look at the problem of gesture recognition for this fixed set of gestures to control drones.

In this paper, we propose a Maximum Entropy Markov Model (MEMM)-based system to recognize **nine gestures relevant to controlling drones** as a first step towards this goal. MEMM is a graphical model used for sequence labeling, by combining features of Hidden Markov Models (HMMs) and maximum entropy models. For building the recognition model, we cluster the frames based on how close they are in terms of joint positions and obtain transition probabilities on the clusters. Next, for inference, given a gesture sample, we compute its conditional dependence graph. We compare this graph against the gesture graphs generated at training time to classify the sample.

We evaluated our system on a public dataset with 21 subjects and used their aircraft marshalling gestures as shown in Figure 1. The proposed model achieves an accuracy of 80% which gives an improvement of 21% over our HMM-based baseline. In addition, we compare our proposed system to three other state-of-the-art ones and it achieves up to 8.1% relative improvement in accuracy.

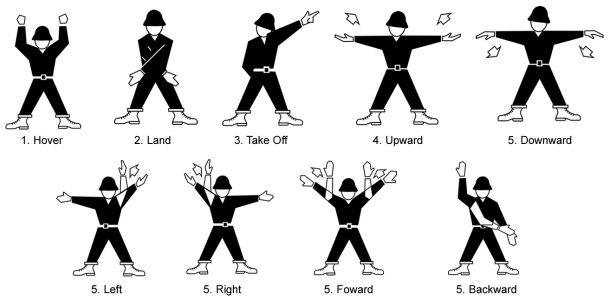


Figure 1. Aircraft Marshalling Gestures.

In summary, our main contributions are:

- We present a two layered MEMM-based gesture recognition drone controller system.
- We use a data driven approach to identify the best features that recognize the gestures and show how it improved the accuracy.
- We evaluate our proposed system against three different models on a public data-set of nine different gestures.

The rest of the paper is organized as follows: Section 2 and Section 3 present our approach in detail. We discuss our experimental evaluation in Section 4. Finally, Section 5 concludes the paper and gives direction for future work.

2. Our Approach

In this project, we employ a supervised-learning approach to recognize user's gestures to control a drone. Figure 2 shows our proposed pipeline to recognize the gestures.

We start by capturing RGB image sequence of the user. Then, we detect their body pose (referred to as skeleton in this paper) and use the different skeleton joints to extract

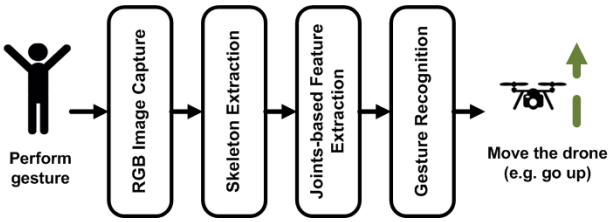


Figure 2. Our gesture recognition pipeline for drone control.

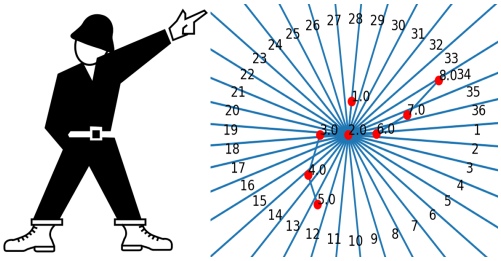


Figure 3. Example of Take Off gesture and its extracted skeleton pose with the tuned baseline bins.

representative features to perform the gesture recognition. We describe the framework's different components in this section.

2.1. RGB Image capture

To identify the user gestures, we can use monocular cameras or RGBD sensors. We believe using a monocular camera is more advantageous to drone control due to its low weight, power consumption, size, and cost as compared to other sensors (Engel et al., 2012). Therefore, we use RGB image sequences from off-the-shelf drone cameras.

2.2. Skeleton Extraction

We extract the user skeleton joints or keypoints with OpenPose (Wei et al., 2016; Cao et al., 2017; Simon et al., 2017), for detection of the human body pose (100 keypoints in total) on each image. Figure 3 shows an example of the extracted filtered joints while performing a "take off" gesture and its corresponding signal from the aircraft marshalling reference (FAA, 2017).

A gesture sample s is a sequence of frames ($s = \{f_1, f_2, ...\}$) and a frame is a set of two vectors X and Y ($f = \{X, Y\}$, each of length 50 values) encompassing the joint's 2D coordinates, e.g. vector $X = \{x_1, x_2, ..., x_N\}$ where N is the number of joints in the extracted skeleton.

2.3. Frames Filtering

Some frames have one or more joints undetected by OpenPose. We consider these frames invalid and remove them. We also reject a sample if more than 10% of the frames are invalid. We reject all samples with very few (< 20) active

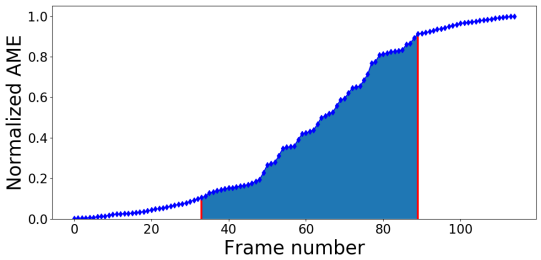


Figure 4. Normalized AME and selected informative frames.

frames (frames where at least one joint position is greater than 0).

In addition, we observe that as a user performs a gesture, there is a temporal evolution of her pose. We can identify 4 groups: neutral (frames without movement before a gesture starts), onset (frames at the beginning of a gesture), apex (frames where the gesture evolves till it is almost done), and offset (superfluous frames after the gesture is complete) (Yang et al., 2014). Gestures are best discriminated/described by their 'onset' and 'apex' segments. Thus, we estimate such frames using their Accumulated Motion Energy (AME) and omit unhelpful ones. We compute the Accumulated Motion Energy (AME) of each frame to measure the distinctiveness of each frame as follows:

$$AME(i) = \sum_{j=1}^{i} |f^j - f^{j-1}|$$

To discard frames corresponding to 'neutral' and 'offset' segments, we remove frames with very low ($< 10\%$) and very high ($> 90\%$) AME values and choose the remaining frames for feature extraction, as shown in Figure 4.

2.4. Joint-based Feature Extraction

We can recognize a user's gesture based on their body pose and arm/hand movement over time. In this module, we extract different joint-based features that manifest the different gestures.

2.4.1. Histogram of 2D Joints

The frame's 2D space is partitioned into b bins with the neck joint as the origin of a Cartesian coordinate system (Figure 3). For each frame, we compute a histogram of 2D joint positions using these b bins. To make the histograms' representation robust against minor variations, joint votes are cast into n neighboring bins using a Gaussian weight function (Xia et al., 2012). This distributes the vote for a joint in a bin across neighboring ones based on how close the joint location is to the boundary of the bin. We selected $\sigma = 2$ ($n = 3$) i.e. current bin + one bin on each side.

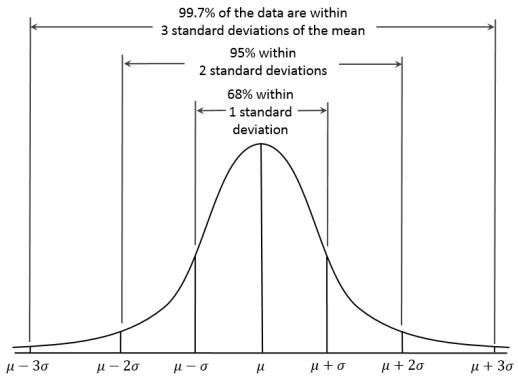


Figure 5. Gaussian bin voting with $\sigma = 2$.

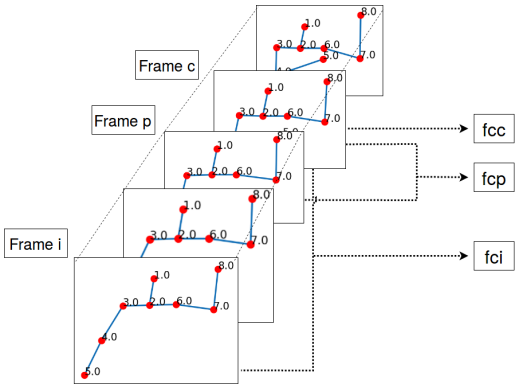


Figure 6. Computation of pose features for Eigenjoints features.

2.4.2. Joints Spatial Displacement

To capture the body movement and how the user's pose changes over time, we extract the following features for each pose: the posture in the current frame-c as the difference of all joint positions in the frame (f_{cc}), the displacement of each joint relative to the previous frame as the difference between all pairs of joints in frame-c and frame-p (f_{cp}), and the position offset from the start of the gesture (f_{ci}) (Yang et al., 2012).

$$f_{cc} = \{x_i - x_j | i, j = 1, 2, ..., N; i \neq j\}$$
$$f_{cp} = \{x_i^c - x_j^p | x_i^c \in X_c; x_i^p \in X_p\}$$
$$f_{ci} = \{x_i^c - x_j^i | x_i^c \in X_c; x_i^i \in X_i\}$$

2.4.3. Joints Orientation Displacement

We extract features based on the joints' relative orientation as another modality for capturing the human body movement. Generally, orientation-based features are invariant to the user position, body size, etc. We first extract features based on spatial orientations of pairwise joints. Specifically, we compute the relative orientation of all pairs of the user's skeletal joints acquired at the same time step t:

$$\theta_{(i,k)} = tan^{-1}(\frac{y_i - y_k}{x_i - x_k}) \quad (1)$$

where $\theta_{(i,k)}$ is the relative orientation between joints j_i and j_k. Additionally, to capture the temporal change in the joints' orientation, we compute the difference between orientations of the same joint across a temporal sequence of frames. Specifically, for a joint i at time t, we compute its orientation displacement relative to the same joint at time $t - 1$ and from the start of the gesture (at time 0).

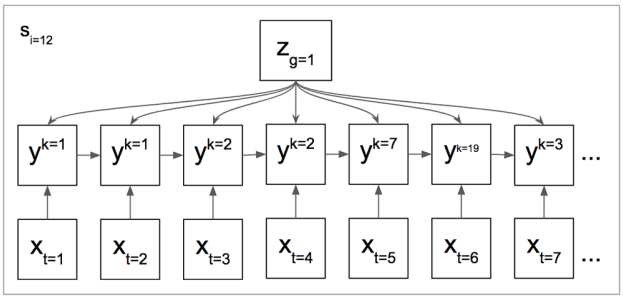


Figure 7. MEMM Model: Graph for a given gesture sample s_i. z_g is the class of gesture g, the y^k's are sub-gestures belonging to cluster k, and x_t's are frames at time instant t.

2.5. Gesture Recognition

Human gestures include a structure as they evolve over time. We capture that through an MEMM-based model. In addition, we employ an on-the-fly graph structure selection to automatically adapt to gesture variations, as typically people perform gestures at different rates. We describe our MEMM model in detail in Section 3.

3. MEMM-based Gesture Recognition

3.1. Gesture Model Learning

We model our gestures using an MEMM graph (McCallum et al., 2000) as shown in Figure 7. To model the graph's probabilities, we learn the transition between poses that characterize each gesture as follows. First, we cluster all frames x in the training data into *sub-gestures* (y) as a coarse partitioning of the poses. Each sequence of sub-gestures is then labeled with the class gesture label z. We compute the distributions that contribute to the graph's probabilities with $m \in \{0, ..., n\}$, where n is the offset from the beginning of the sample as follows:

- $P(y_t|x_t)$: This models the probability of sub-gesture y_t given features x_t.
- $P(y_{t-m}|y_{t-m-1}, z_g)$: This models the probability that a sub-gesture y_{t-m-1} transitions to sub-gesture y_{t-m} if the user is performing gesture z_g

3.2. Inference

Given a new sequence of N frames, the algorithm predicts the gesture z the user is performing by computing the graph G_N with the highest probability. This can be done through dynamic programming (Sung et al., 2012). See Appendix B for the dynamic programming algorithm.

3.3. Hyperparameter tuning

Two important set of parameters affect the algorithm's accuracy: the number of sub-gestures, and which raw features from x to observe. We estimate that on average, each gesture contributes three distinctly new sub-gestures. Therefore, in our implementation, we cluster the frames into 27 sub-gestures. Further experiments on training data confirms that this is an optimal choice.

Additionally, it is important to select the joint coordinates that differentiate the gestures well. Using the entire set of 100 coordinates is both distracting and computationally expensive. As a simple heuristic, we find the best coordinates to track, by finding one single coordinate that best distinguishes each pair of gestures. We select the top 26 coordinates whose transitions between frames best distinguish gestures from one another in our training dataset.

4. Evaluation

In this section, we discuss our system evaluation. We use a public data-set (Wan et al., 2016) which consists of different gestures performed by 21 subjects—we use the aircraft marshalling gestures. The data-set provides three splits: training, validation and testing. We train the models using the training split and tune the hyperparameters using the validation set. Then, we measure the accuracy, precision, recall, F-measure and the confusion matrix of the models on the test-set. We compare the performance of our proposed model to an HMM-based baseline (Xia et al., 2012) (Section 4.1). In addition, we compare with two other models (Section 4.2): (1) A Naive-Bayes Nearest Neighbor on EigenJoints (Yang et al., 2014) and (2) A Multilayer Perceptron (MLP)-based model. Due to page constraints, we describe only the configurations that gave the best results and we report those.

4.1. Baseline

Our baseline is based on the approach presented in (Xia et al., 2012) which uses a Gaussian HMM. Using the histogram of join positions described in section 2.4.1 as the training data, we cluster the histogram vectors into K clusters (a K-word vocabulary). Each gesture is then represented by a series of visual words. We use separate HMMs to construct a model for each of the drone-control gestures. To classify a new gesture, we take its visual words sequence and calculate the score for this sequence with every gesture model. Then, we recognize it as the gesture with the highest probability model.

We evaluate the baseline, with configuration from (Xia et al., 2012), $K = 125$, HMM states $N = 6$, with bins $b = 14$, and histogram Gaussian voting over bins $n = 3$. We further tuned these parameters on the validation set. The configuration of $K = 200$, $N = 40$, $b = 36$, and $n = 3$ gave the best results on the test set reported in Section 4.3

4.2. Other Comparison Models

For the comparison models, we focus only on the upper body joints with 8 keypoints (head, neck, and shoulder, elbow and wrist for both sides) as other keypoints are either not practical or not part of the gesture. This is as they performed better with spatial and orientation-based features as explained in the section.

4.2.1. EigenJoints-based gesture recognition

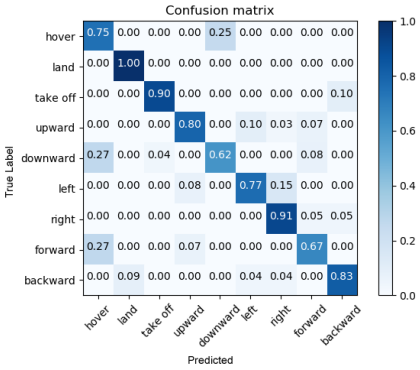
We employ the joints spatial-displacement features described in Section 2.4.2. To reduce noise and redundancy in data representation, we apply Principal Component Analysis (PCA) for dimensionality reduction of each feature vector, and use the M most significant features. We have empirically decided through multiple trials that $M = 16$ suffices. For the gestures recognition, we used Naive-Bayes-Nearest-Neighbor (NBNN) classifier (Yang et al., 2012) on the feature vectors of each gesture sample. NBNN is a simple classifier, which helps compute the *sample-to-class* distance. The sample-to-class distance and the subsequent class C^* which best describes the gesture sample are computed as:

$$C^* = \underset{C}{\operatorname{argmin}} \sum_{i=1}^{M} ||d_i - NN_c(d_i)||^2$$

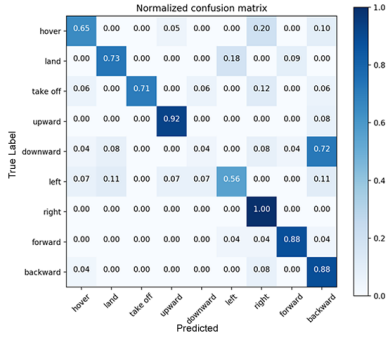
where d_i, $i = 1, 2, ..., M$ is an EigenJoints descriptor of frame-i in a testing sample; M is the number of frames; $NN_c(d_i)$ is the (single) nearest neighbor of d_i in class-C; and, C^* is the optimal class label given to the test sample. We use balltrees to store frame descriptors corresponding to each class for efficient NN computation.

4.2.2. Multilayer Perceptron

We train a Multilayer Perceptron using backpropagation on the joints spatial and orientation displacement features (Section 2.4.2 and Section 2.4.3). The neural network has a ReLU activation and the log-loss function is optimized using a stochastic gradient-based optimizer (Kingma & Ba, 2014).



(a) The proposed model



(b) HMM (*tuned baseline*)

Figure 8. Our proposed model and the HMM (*tuned baseline*) confusion matrices.

4.3. Evaluation Results and Error Analysis

Table 1 compares the accuracy, gestures average precision and recall for our proposed model to the baseline and other comparison models. We can see that our model has an accuracy improvement of 21% to our baseline and up to 8% with the comparison models.

The confusion matrix in Figure 8(a) shows that the data-driven features and the proposed MEMM model helped reduce the misclassification of gestures as compared to the tuned baseline (Figure 8(b)). For instance, the baseline model mislabels "downward" as "backward" very frequently, while the MEMM does not have this problem. This is due to the data driven features MEMM uses. However, we can see that while the MEMM have an overall better performance, on some gestures, the HMM outperforms it.

5. Conclusion and Future Work

We proposed a gesture recognition framework for drone control. It employs a modified MEMM to capture the user's gesture structure and variation. Our model uses a data driven approach to extract the best discriminative features and employs an on-the fly graph structure selection to recognize the user's gesture. We achieved an accuracy of 80% which has a relative improvement of 21% over HMM baseline and more than 8.1% over other comparison models. From our data analysis, we can see that combining multiple probabilistic graphical models would further improve the accuracy. There is also reason to believe that using the hand-joints with HMM models can help improve their accuracy. Additionally, using each joint as a reference point for binning would give us more information about relative movement.

Table 1. Summary of the evaluation results comparing our approach to the baseline and other comparison models.

Model	Precision	Recall	Accuracy
MEMM	**0.80**	**0.80**	**0.80**
Multilayer Perceptron	0.77	0.77	0.74
Eigenjoints	0.80	0.79	0.76
HMM tuned	*0.67*	*0.71*	*0.66*
HMM (Xia et al., 2012)	0.32	0.37	0.38

Contribution

Each member of the team took a lead in implementing different parts of the work: Phil worked on the MEMM, Ananth worked on the EigenJoints model, Heba worked on the MLP, and Carolin and Snehesh worked on feature extraction and the HMM. We met regularly and all contributed to the final writeup.

APPENDIX

A. Related Work

There is a large body of previous work on human gesture recognition for various use cases. In this section, we discuss related models and highlight how our model differs.

Gestures are an important aspect of human robot interaction that allow us to not be restricted to traditional joystick or screen based remote controllers. Work such as gesture and voice controlled tour guide robot (Alvarez-Santos et al., 2014) and gestural interaction with robotic arms to handle dangerous liquids (Ende et al., 2011) are such examples. While there has been work in trying to control robots by recognizing natural human gestures between humans (Cauchard et al., 2015), there is no concrete universal gestures that a robot could learn across human diversity. So this requires a very large sample from varying sizes, shapes, viewpoints, speeds, cultures etc which is very difficult. We leverage the aircraft marshall gestures to be recog-

nized by our drones to control them. In this context, hands or upper body motion are primary features in recognition of gestures using various models that we have explored in this paper (Vemulapalli et al., 2014; Xia et al., 2012; 2013; Yang et al., 2012; Sung et al., 2012; Cao et al., 2017; Simon et al., 2017; Yang et al., 2014). There have been research work using hand motion to recognize gestures as alphabets (Elmezain & Al-Hamadi, 2007) and recognize Sign Language (Starner & Pentland, 1997). Gesture extraction and recognition has also been done for human-robot interaction (Yang et al., 2007; Cauchard et al., 2015).

B. Pseudocode for MEMM key procedures

The Python-like code snippets below show key procedures in our MEMM model. Specifically, `predict` shows the dynamic-programming algorithm that predicts the most probably gesture by efficiently searching for the most probable graph that explains the frame sequence. `distinguishing_keypoints` shows how to select the most distinguishing key-points, each excelling at telling apart one specific pair of gestures in the training data.

```
def predict(frames):
  Pr[t][g] ← 1/num_gestures
  for t ∈ [0, frames.length):
    for g ∈ gestures:
      Pr[t][g] ← max_{t1<t,g1} (Pr[t1][g1] × p(t1, g, t))
  return majority_t (argmax_g Pr[t][g])
```

```
def distinguishing_keypoints(X,Y):
  return
    {distinguishing_keypoint
      (y1,y2,filter(X,Y,labels={y1,y2}))
     for y1, y2 ∈ gestures
     if y1 < y2}

def distinguishing_keypoint(y1,y2,X,Y):
  def score(k):
    Xk ← project(X, k)
    c ← Classifier(data=Xk,labels=Y)
    return count(Y == c.predict(Xk))
  return max(all_keypoints, key=score)
```

References

Alvarez-Santos, Víctor, Iglesias, Roberto, Pardo, Xose Manuel, Regueiro, Carlos V, and Canedo-Rodriguez, Adrián. Gesture-based interaction with voice feedback for a tour-guide robot. *Journal of Visual Communication and Image Representation*, 25(2): 499–509, 2014.

Cao et al., Zhe. Realtime multi-person 2d pose estimation using part affinity fields. In *IEEE CVPR*, 2017.

Cauchard, Jessica R, Zhai, Kevin Y, Landay, James A, et al. Drone & me: an exploration into natural human-drone interaction. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pp. 361–365. ACM, 2015.

Elmezain, Mahmoud and Al-Hamadi, Ayoub. Gesture recognition for alphabets from hand motion trajectory using hidden markov models. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pp. 1192–1197. IEEE, 2007.

Ende, Tobias, Haddadin, Sami, Parusel, Sven, Wüsthoff, Tilo, Hassenzahl, Marc, and Albu-Schäffer, Alin. A human-centered approach to robot gesture based communication within collaborative working processes. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3367–3374. IEEE, 2011.

Engel et al., Jakob. Camera-based navigation of a low-cost quadrocopter. In *IROS*. IEEE, 2012.

FAA. Airmarshal signals. `https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/airplane_handbook/media/04_afh_ch2.pdf`, 2017. Online; accessed 12 December 2017.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

McCallum, Andrew, Freitag, Dayne, and Pereira, Fernando CN. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pp. 591–598, 2000.

Simon et al., Tomas. Hand keypoint detection in single images using multiview bootstrapping. In *IEEE CVPR*, 2017.

Starner, Thad and Pentland, Alex. Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pp. 227–243. Springer, 1997.

Sung et al., Jaeyong. Unstructured human activity detection from RGBD images. In *IEEE ICRA*, 2012.

Vemulapalli et al., Raviteja. Human action recognition by representing 3D skeletons as points in a lie group. In *IEEE CVPR*, 2014.

Wan et al., Jun. Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In *IEEE CVPRW*, June 2016.

Wei, Shih-En, Ramakrishna, Varun, Kanade, Takeo, and Sheikh, Yaser. Convolutional pose machines. In *CVPR*, 2016.

Xia et al., Lu. View invariant human action recognition using histograms of 3D joints. In *IEEE CVPRW*, 2012.

Xia et al., Lu. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In *IEEE CVPR*, 2013.

Yang, Hee-Deok, Park, A-Yeon, and Lee, Seong-Whan. Gesture spotting and recognition for human–robot interaction. *IEEE transactions on Robotics*, 23(2):256–270, 2007.

Yang et al., Xiaodong. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *CVPRW*. IEEE, 2012.

Yang et al., Xiaodong. Effective 3d action recognition using eigenjoints. *J. of Visual Communication and Image Representation*, 2014.