CS5340 Group13
Hand Pose Estimation from Single Depth Images

Yang Xuan	**A0159066X**
Liu Zhendong	**A0159369L**
Huang Mengying	**A0159188M**

1 Overview

This program implements Random Forest to do prediction for hand pose from single depth images, referencing the methods used in [1]. We have finished the Baseline-M that stated in [1]. Our dataset is downloaded from http://hpes.bii.a-star.edu.sg/index.php/instructions, which contains both training dataset and test dataset, 15 sub-directories respectively. Each of them contains information pertaining to various hand poses for different subjects. The goal of this report is to introduce what our team has done and propose what we plan to do in the future month. We will begin with the design of this program to show our understanding on the principles and our design to implement. We will continue by specifying the process of our implementation. Then, we will give a short conclustion to state what we have accomplished. Further, we will discuss about our future work.

2 Design

We divided our systems to three parts: preprocess, model training, prediction.

2.1 Preprocess

Our system adopts hand patch instead of individual pixel as one training examples. The original depth image in dataset contains background and other objects which are irrelevant to our hand pose estimation tasks, such as brackets of the data-glove device. Therefore, we need to obtain a hand-centered bounding box after preprocessing the image. We first remove the background and noisy points by discarding the pixels of which depth value is greater than a certain threshold. Which means if a pixels depth is greater than the threshold, we directly set its value to be 32001 which is the maximum depth value. Then we fixed the bounding boxs size to be 90×60 and slide the windows through whole images to obtain the bounding box which contains the most non-background pixels. Here are some examples of image cropping.

2.2 Model training

We need to train a lot of decision trees for random forest and each tree has the same training process, so we only describe the procedure of training a decision tree.

2.2.1 Depth features

After preprocessing, each hand pose can be represented by following form: $E(X, Y)$, where X is a 90×60 dimension vector and Y is a 20×3 dimension vector. Each dimension in X is calculated by . L is the 20 joints 3D position. So our goal is building a model based on the training examples and predicting the test examples Y according to the corresponding X.

2.2.2 Split criteria

At each split node, we randomly choose a feature subset. For each feature, trying different split value to calculate information gain. Finally, we choose the best feature and split value which result in largest information gain to split this node into two children nodes. We only consider quantiles as our split value for the sake of efficiency. As for Entropy which is used to calculate information gain of a node, we use the entropy of multivariate Gaussian distribution like [1]. When calculating the Gaussian distribution, we add small random numbers to the diagonal of matrix to make sure it is invertible, which means the determinant of the matrix can never equal to 0.

2.3 Prediction

Prediction of random forest is simply obtained by averaging the prediction from trees. So we only introduce how to obtain prediction from a decision tree here. Assume we represent an example as e(X,Y) .X is the feature vector and Y is the label vector. we let test example reach down to a certain leaf node L according to corresponding split features and split values which are selected during the model training phase. In current baseline model, we directly average all the training examples label vector in the leaf node L to generate prediction. But we have implemented the image similarity function which is based on the pixel gradient information. Later on, we can use this kind of similarity as vote weight for each training example in order to improve prediction accuracy.

3 Implementation

Conclusion

We implemented our program in two modes, namely, debug and real time mode. In debug mode, we split the training data into two sub-datasets, 70% for training model and 30% for predicting. While in real time mode, we used the whole training data to train the random forest model to improve the prediction accuracy. Up to now we have trained each decision tree for the random forest successfully. Besides, in the random forest, the vote is done by calculating the average value of all the results and uses average value of labels in each leaf node to do prediction, which is also referred to as Baseline-M.

Future Work

Instead of making prediction with the empirical average as in Baseline-M, we will implement other methods to improve our prediction accuracy. As mentioned in [1], Baseline-S, during which static weights will be assigned to each training examples, but it remains unchanged during prediction stage. So, we would adopt a dynamically weighted scheme (i.e. DHand), where each of the weights can be decided at runtime. In depth image, the direction of the gradient indicates the higher depth value, so we can use gradient information to represent each pixel in an image. In the improved similarity computed function, sufficient statistics will be used to fully represent the raw input image hand image data, then execute bitwise OR operation to calculating the similarity score. Compared to empirical mean method, this dynamical weighted way can save huge storage memory because it does not need to store huge raw images. In the future month, we are going to improve our program from Baseline-M to DHand scheme. Meanwhile, we will make efforts to optimize our algorithms to speed the running time.

References

[1] Chi Xu, Ashwin Nanjappa, Xiaowei Zhang, Li Cheng. *Estimate Hand Poses Efficiently from Single Depth Images*. In In International Journal of Computer Vision (IJCV), 2015.

[2] Conrad Sanderson and Ryan Curtin. *Armadillo: a template-based C++ library for linear algebra*. Journal of Open Source Software, Vol. 1, pp. 26, 2016.