Human Detection, Tracking and Activity Recognition from Video

Mihir Patankar
University of California San Diego
San Diego, USA
mpatanka@eng.ucsd.edu

Abstract—Human detection, tracking and activity recognition is an important area of research with applications ranging from entertainment, robotics, surveillance and elderly care. In this project, I implemented and developed various human detection and tracking algorithms using Robot Operating System (ROS) in Linux and using MATLAB R2016b in Windows. Implementing the algorithms using Kinect and MATLAB, real-time human detection and skeleton tracking was achieved with 30 fps. The second part of the project was to collect data to implement various machine learning algorithms to predict human activity. Currently, the system detects 3 activities with 96.02% accuracy using Decision Trees for one human.

Keywords—Human detection, skeleton tracking, activity detection and classification

I. INTRODUCTION

Human body detection and tracking algorithms have faced various challenges because of different human poses, illuminations, complex backgrounds and occlusion. Also, given the importance of human detection in applications like self-driving cars, elderly care, human robot interaction, there is a need to perform this task accurately and in real time. Failure to do so can lead to undesirable consequences. Hence, a lot of research has been focused on developing algorithms which can not only detect and track humans accurately but are able to execute in real time.

There has been extensive research on activity recognition using ubiquitous wearable sensors such as accelerometers, IMU’s and gyroscopes on various body parts. However, this approach is intrusive and requires continuous maintenance of the sensors. A solution to this is to use non-obtrusive commercially available imaging sensors for human tracking and activity recognition. This approach has its own drawbacks as well. The performance of imaging sensors suffers with changing lighting conditions. Also, continuously monitor an area and thus might invade the privacy of the individuals. Another issue is that these sensors cannot be attached to a particular user and thus continuous monitoring might be difficult.

Commercially available low cost depth sensors solve this issue by using depth maps which are invariant to illumination. In this project, we use data only from the depth sensor thus solving the issue of privacy and plan to fuse data from multiple imaging sensors in future which can cover the entire area of the environment where the user needs to be tracked.

In this work, human detection and tracking is performed using Microsoft Kinect. Joint data returned by the Kinect is used as training data for various human activities. Activity classification is performed using Machine learning algorithms like k-nearest neighbours, multi class SVM and Decision Trees available in the MATLAB Statistics and Machine Learning toolbox and a comparative study of them is presented.

II. RELATED WORK

Histogram of Oriented Gradients (HOG) [1] is similar to edge orientation gradients along with a linear SVM classifier for speed and simplicity. This method gives near perfect results on MIT pedestrian dataset. Since the MIT dataset contains a limited set of poses and orientations, the authors came up with a more challenging dataset called INRIA. This method had a simpler architecture and gave better accuracy than the other methods available at that time. It captures the edge orientations which gives the local shape and still being invariant to the geometric and photometric features. The crucial aspects of HOG detectors are that it extracts information from abrupt edges at the finest available resolution of the image and that local contrast normalisation improves detection results. This system performs better than the best available detector using Haar features.

This paper [2] gives a fast and robust people detection algorithm for mobile robots. They use voxel grid filtering which is downsampling of the RGBD data to achieve real time performance. They also have robust human detection even in cluttered environments using sub-clustering. For tracking, they use inputs from detection modules and solve the data association problem to predict the next pose. ROS implementation of their algorithm on a standard computer achieves 26 fps.
Use of five accelerometers worn by the user on different body parts to detect activities [3] was demonstrated in this work. This data was collected from 20 users performing various activities throughout the day. Classification algorithms were implemented on this dataset extracting features using FFT from which decision trees gave the highest overall accuracy of 84%. Increasing the number of accelerometer sensors on the body improves the performance. However, accelerometers are obtrusive since users have to wear and maintain the required sensors.

Another work [4] uses K-means and SVM for detection and classification of postures and finally HMMs for activity recognition. They use a subset of joints discarding the redundant or noisy data to be used as features. They obtain accuracy rates from 80-100% for the 18 activities classified using their method. They have also introduced the KARD activity dataset containing 18 different activities consisting of various gestures and actions.

Section 3 describes the specifications and advantages of Microsoft Kinect over other available imaging sensors. Section 4 describes human detection and tracking algorithms developed in ROS. Section 5 gives an overview of implementation of the algorithms in MATLAB R2016b. Section 6 and 7 explains the data acquisition and implementation of machine learning algorithms for activity classification.

III. MICROSOFT KINECT
Kinect is a set of motion sensing devices developed by Microsoft for Xbox gaming consoles. It consists of an RGB Camera, an infrared (IR) emitter and an IR depth sensor to calculate the depth image. It is particularly suitable for indoor environments. Some important kinect specifications [5] are given below.

- RGB 1280x960 at 12fps
- RGB 640x480 at 30fps
- IR 640x480 at 30fps
- Depth 640x480, 320x240, 80x60 at 30fps
- 43° vertical by 57° horizontal field of view
- 27° vertical tilt angle

Taking into consideration the above features, kinect provides significant advantages over monocular RGB cameras. For instance, the depth stream provided by kinect is independent of intensity and thus more robust. Thus, taking into consideration the reliability, low cost, availability and software support, kinect was the best choice of sensor used for the project.

IV. IMPLEMENTATION USING ROS
Robot Operating System (ROS) is a framework designed with large-scale software integration to meet the needs of the ever growing Robotics community [6]. It is a collection of tools and libraries that help simplify the task of building collaborative complex software modules for robots.

The major advantages of ROS are that it facilitates fast wireless communication between different onboard and offboard processors, allows use of multiple programming languages, code reuse and is free and open source. ROS uses a specific nomenclature where nodes are software modules and they communicate through messages. These messages are published on topics which are subscribed by the nodes which read them. Finally, services are used for asynchronous communication.

Assuming the environment has only 1 human and no occlusion, a naive implementation using depth segmentation was performed in ROS Indigo with Ubuntu 14.04. This method could accurately give the x, y and z coordinates of the human under the restrictions stated above.

However this wasn’t a particularly robust solution, thus another implementation was performed using point clouds and SVM. Depending on size of point cloud, it tries to associate a previously seen human to the current human detected. The point cloud is passed as an input to a pre-trained SVM which determines if the point cloud represents a human or a non-human. The tracking is stopped if the object is stationary for 5000 frames which is a tunable parameter. This node publishes information on a topic called “people” which consists of the number of people tracked and their x, y and z world coordinates with respect to the kinect. Figure 1 shows the human being tracked and labelled using a ROS tool called “rviz” and the number of humans and their location being published. This method gave a lot of false positives and low frame rate due to the ROS overheads and intensive computations.

![Figure 1](image)

V. IMPLEMENTATION USING ROS
MATLAB R2016b has support for Kinect v1 hardware using their image acquisition toolbox. The kinect color sensor and the depth sensor have their own device ID in the MATLAB environment. The kinect returns four data streams [7] namely image stream through the color sensor in various formats like
RGB and YUV, depth from disparity for each pixel from depth sensor, skeletal data from depth sensor and metadata about each skeleton tracked. Audio stream is returned from the speaker array and can be used with the MATLAB AudioRecorder module.

The skeleton data was most useful for human detection and tracking and the metadata was used for activity classification. Kinect has the ability to segment 6 people of which 2 can be actively tracked and the remaining four can be passively tracked. The metadata returned for skeletons contains timestamp, overall position of the skeleton along with image and world coordinates (in meters) of 20 joints.

The kinect can be used to track position (coordinates of hip) or skeleton (coordinates of all joints) in standing (returns 20 joints) or seating (returns 10 joints) positions.

a. IMPLEMENTATION ON IMAGES

Initially skeleton tracking was implemented on images synchronised from the color and depth sensors. For this, the tracking mode is set to manual and triggering to manual to trigger the depth and color sensors at the same time. The frames per trigger are set to 100 to allow enough time for synchronisation of the color and depth sensor. Any frame can be arbitrarily chosen from the 100 acquired frames to extract information. The getdata function gives all the required data from both the sensors. Depth sensor returns metadata for each frame which contains the joint and world coordinates for the body being tracked.

Once the coordinates are returned, a custom skeleton viewer function can be used to visualise the skeleton data. The joint image coordinates, the color image and the number of skeletons being tracked can be passed to a function which can then display the skeleton data superimposed on color image. The results of the implementation are shown in Figure 2 below.

b. IMPLEMENTATION ON VIDEO

The approach explained above can be extended for video using frames per trigger as 1. However, frequent calls to the skeleton viewer function slows down the code and makes it difficult to execute in real time.

A simple modification of the above approach completely removing the color data and overlaying the skeleton data on the depth image instead of calling skeleton viewer function allows the code to execute in real time. The resulting image is shown in Figure 3. The timing information for both the codes returned by MATLAB is as shown in figures 4 and 5.

This can also be used to detect interaction with objects based on proximity. The world coordinates of an object can be saved in the code and position of the human can be compared with that of the object to detect interaction. This method however, has the drawback that it assumes kinect and the object are at fixed positions.
VI. DATA COLLECTION FOR ACTIVITY CLASSIFICATION

The training data for activity classification was obtained using real-time skeleton tracking with MATLAB R2016b as described in section 5. The data collected represents three types of activities namely standing, waving right hand and waving left hand labeled as 1, 2 and 3 respectively.

Each sample consists of 12 features. As described in section 5, using the kinect sensor we obtain a set of 20 joint world coordinates which are used as features. The features consist of x, y and z vector coordinates for left hand, right hand, left leg and right leg considering hip as the origin. Thus the x, y and z coordinates of hip are subtracted from the coordinates of left hand, right hand, left leg and right leg before using them as features.

The training data consists of 771 samples out of which 270 are for activity 1, 296 for activity 2 and 205 for activity 3. The testing data is obtained independently using procedure similar to that for training data. It consists of 427 samples out of which 179 are for activity 1, 129 for activity 2 and 119 for activity 3. The data collected was stored in two comma separated values file format.

VII. ACTIVITY CLASSIFICATION

The machine learning algorithms described below were trained with the training data called data_train and labels called labels_train obtained in section 6. The test data called data_test was used to make predictions. The predicted results were stored in a matrix called prediction. The predicted values were compared against the ground truth labels called test_labels.

A. ACCURACY CALCULATION

For accuracy calculation, difference was calculated between prediction and test_labels matrices and stored in accuracy matrix. The correct predictions would result in zero and incorrect predictions would give a non-zero result. Thus to calculate the accuracy percentage, the number of zeros were calculated from the accuracy matrix.

B. FEATURES USED

Scatter plot for various features were observed and only the features with maximum variance for each label were used instead of all features. The scatter plots for x, y and z coordinates of left hand and right hand can be seen in figure 6, 7 and 8 respectively. It can be observed that figure 7 has the maximum variance for the three labels. Thus the results were obtained once using all features, then using only y coordinates of hands which results in a much better prediction accuracy for most classifiers.
C. ALGORITHMS USED
The following classifiers were used from the MATLAB R2016b Machine learning and Statistics toolbox.

a. K-NN
The principle behind k-nearest neighbor algorithms is to find k training samples closest in distance to the test sample, and predict the label based on majority vote. The number of samples k is user defined and the distance can be any measure but generally squared euclidian distance is used as a metric.

The prediction accuracy for the classifier is given in Table 1 for three different values of k. A comparative graph is shown in figure 9. Thus, we can see that the prediction accuracy increases when we use only a subset of features with maximum variance.

<table>
<thead>
<tr>
<th>k</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All features</td>
</tr>
<tr>
<td>3</td>
<td>77.29</td>
</tr>
<tr>
<td>5</td>
<td>78.69</td>
</tr>
<tr>
<td>7</td>
<td>77.75</td>
</tr>
</tbody>
</table>

b. MULTI CLASS SVM
Support Vector Machines (SVM) are discriminative classifiers which output the optimum hyperplane that separates the training data. Margin is defined as the distance of data from the separating hyperplane. The objective of SVM is to maximize the margin of the training data.

The prediction accuracy using Multi class SVM is shown in Table 2. It can be seen that it is almost the same for the two types of feature sets used.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>All features</td>
<td>71.43</td>
</tr>
<tr>
<td>Subset of features</td>
<td>70.02</td>
</tr>
</tbody>
</table>

c. DECISION TREES
The goal of Decision Trees is to generate a model that predicts the classification labels based on decision rules based on the training data. At each level of the tree, the model tries to maximize information gain by considering all possible splits based on the features used.

Figures 10 and 11 show the decision trees generated in MATLAB R2016b using all and subset of features respectively. The prediction accuracy using decision trees is given in Table 3. Figure 12 shows a comparison of accuracy using all features and a subset of features.
human detection and tracking accurately. For activity classification, various machine learning algorithms were implemented like k-NN, multi class SVM and decision trees of which decision trees gave a prediction accuracy of 96.02% on the test set. Future work will be to combine object recognition and activity detection to detect human interaction with objects.

REFERENCES

[7] Mathworks Documentation, Key Features and Differences in the Kinect V1 Support

Table 3

<table>
<thead>
<tr>
<th></th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>All features</td>
<td>95.08</td>
</tr>
<tr>
<td>Subset of features</td>
<td>96.02</td>
</tr>
</tbody>
</table>

VIII. CONCLUSION

In this work, human detection and tracking algorithms were implemented using ROS and MATLAB R2016b using the Microsoft Kinect sensor. While the ROS implementation had a low frame rate and a high rate of false positives, MATLAB implementation was able to perform at 30 fps giving real time