Real-Time Driver Activity Recognition with Random Forests

Lijie Xu and Kikuo Fujimura
Honda Research Institute USA
425 National Ave.
Mountain View, CA USA

ABSTRACT

In this work, we introduce a real-time driver activity recognition method which takes a sequence of depth images as input and outputs an activity class among a pre-determined set of driver activities. A classification algorithm called Random Forests is employed and further enhanced by a unique state based inference system to reduce initial classifier errors. For example, frequent changes in driver activities are penalized so as to stabilize the output. The cost of activity change is decided by a state inference system which takes both temporal and spatial coherence into account. The paper will introduce the training system, explain the state inference system and the cost based penalty calculation. Finally we will discuss the results and future work.

Author Keywords
The Random Forest, State Inference Enhancement, Driver Monitor, Driving Activity Monitor.

ACM Classification Keywords
I.5.4 [Pattern Recognition]: Application – computer vision

INTRODUCTION
Nowadays, the automotive user interface has become an important research area – intuitive and effective communication between the driver and the vehicle is important to build better driving experiences. Communication beyond a car and its driver to an extensive network of cars and their drivers on the road can help build an autonomous driving system [1].

In this paper we introduce a hands-free Auto UI framework that allows the user to interact with the vehicle without interference of their driving activities. Users' actions are captured by a depth camera placed on the dash board and streamed to the system. The Random Forest framework is used to train and classify the input to one of the predetermined user activity categories. The recognition result of the Random Forest is fed into a post processor state machine to infer possible activities based on the history. The Random Forest output is corrected based on temporal and spatial coherence.

AutomotiveUI'14, September 17 - 19, Seattle, WA, USA.
Copyright (c) 2014 ACM 978-1-4503-3212-5/14/09…$15.00
http://dx.doi.org/10.1145/2667317.2667333

The Random Forest (hereafter, referred to as RF) is a decision tree based ensemble training and recognition algorithm. The major idea is as follows. Each decision tree in the forest is trained by a randomly selected set of tests. The collection of relatively weak tree classifiers then vote for the final result. The major advantages of RF are its robustness, its speed, and accuracy. Another important merit of RF is that it is not sensitive to the notorious over-fitting problem. Process speed is important because the goal is to build a real-time recognition system. And, because the complexity of the data, which are 2D images collected from different users, over-fitting must be avoided so the trained model can be applied to users not included in the training set. Though the Random Forest is considered one of the most precise classifiers, like many other training based classification methods, it also has misclassification problems, that is, recognition errors.

The driver usually has a relatively limited set of activities while driving and is subject to certain restrictions on transits between different behaviors; this provides the opportunity to improve recognition results based on temporal and spatial coherency. We propose a state based inference system, as a post processing procedure to correct the RF's recognition errors. For simplicity, from this point on, we call our state based inference system the state machine. The state machine consists of a collection of states. Each state represents a confidence value of one of the activity categories from the predetermined collection. Change from one driving activity to another driving activity in time is modeled as changes in confidence values (state transit) in the state machine. The main idea of the state machine is to penalize a state transit based on temporal coherence and physical likelihood of the involved states. After injecting a new state, the state machine updates the confidence value of each state. The state machine is queried to get the inference result state (the one corresponding to the largest confidence value), which can be different from the injected state.

Temporal restriction suppresses identification errors under the assumption that a very short sequence of one action is rare in real life. For example, adjusting radio usually cannot be accomplished by less than 3 frames. This is accomplished as follows. Each state in the state machine maintains a floating point value in [0,1], a dynamic level of confidence. The confidence level of all states changes when the next action is seen by the state machine. The confidence level of a state increases when input includes consecutive frames with the same class. The state with the highest confidence is selected as the state machine output and is the state inferred result.

Spatial coherence reinforces the physical restriction of human activities. The transit between some driving actions is more feasible than transit between other actions. The possibility of transit between states is modeled by a state transit graph. The directed graph represents the probability of a transit using nodes and edges. These states are divided into several groups. One node represents a group of states which can include a single state or a collection of states. The edges represent the probability of transit from a state from one group to a state to another group. A transit between states in the same group has a non-zero probability. Transit among different groups is allowed when there is a direct edge connecting them. Transition cost among groups that do not have an edge connecting them is considered high. The transit graph helps to fine-tune confidence level updates when a transit happens. Transit of low probability is suppressed by a slower reduction rate of the confidence level for the current state, while transits of high probability are promoted by a faster increase rate of the confidence level of the new state.

The rest of the paper is organized as follows. We will first focus on data acquisition and training data preparation. We then briefly discuss the Random Forest training process with parameter selection followed by an introduction to temporal confidence and physical likelihood based state machine concept. Finally, we present the result and conclude the paper by discussions and future works.

RELATED WORKS

Driving activity monitoring has been an active research field. Earlier research works have been concentrated on partial driving activities. Baker et al [2] propose a real time head tracking system that monitors the driver's mental state such as drowsy, alert, aggressive, comfortable and more. Ji and Yang present [3] an eye lid, gaze movement and face orientation to infer the driver's alert level. Instead of using a single feature, the method combines different features such as pupil tracking, face orientation estimation to improve precision and robustness. Doshi and Trivedi [4] study different features' correlations to a future lane change activity. Combinations of these features are fed into classifier and study shows head dynamic is a reliable source to predict the driver's intent to change lanes. Additional information such as eye gaze does not improve the performance significantly.

Zhu and Fujimura [5] propose and compare two head pose estimation algorithms: principal component analysis (PCA) and 3D motion estimation and uses depth images to avoid visual clutter. Murphy-Chutorian and Trivedi [6] provide a survey of head pose estimation research. Besides recognizing partial activities, estimating complete driving activities has been a popular research area as well. Veeraraghavan et al. [7] propose a computer vision aided driving activity system that recognizes safe and unsafe driving activities using a two-step clustering framework. They propose a fast algorithm that combines the two clustering steps to ensure real time performance. In this paper, we provide a driving activity estimation system using in-car depth images. The result provides more comprehensive understanding of the driver's activity compared to partial activity estimation. The challenge is the process speed given the amount of data involved. And the classifier needs to be robust and avoids the over-fitting problem.

Methods based on Random Forests [8] have been widely applied in image understanding. Many applications use RGB images as training and testing data. Criminisi et al. [9] use Random Forests on medical CT data to detect anatomy. Regression forests are used to solve a non-linear regression problem which maps between voxels and anatomy locations, with the criteria of maximize the confidence of the predictions. Gall et al. [10] use them to create a general purpose Hough transform of high performance. The algorithm is validated through published benchmarks. Huang et al. [11] propose a robust and precise real-time head pose recognition algorithm using Random Forests that outperform traditional algorithms such as PCA and LDA and machine learning methods.

Recently, RF has been used on range data by Fanelli et al. [12] to estimate human face normal directions using depth images. Gesture identification is a well-developed research area. Shotton et al. [13] use random forest for gesture identification. Random forest classifier is used to localize joint positions based on per pixel classification. Shakhnarovich et. al. [14] proposes a hash function learning method to estimate input gesture in a timely manner. Agarwal and Trigg [15] describe a nonlinear regression method that recovers human body pose directly from single and monocular images. The paper proposes to take depth image as input and use RF to classify body gestures. The paper defines a test function for decision tree training and information theory based cost function. We adopt test and cost function similar to ones used in [12] and apply them on driver activity recognition, using range data produced by depth sensors. The details are discussed in the following sections.

DATA ACQUISITION

System Setup

A depth camera is placed on the vehicle's dashboard which streams depth images of the driver to the computer. The driver (subject) is asked to pose for the following seven activities: driving normally, reaching for center compartment, reaching for glove compartment, reaching for overhead, adjusting radio, talking on phone, and texting. At the training stage, one video contains the pose of a single driving action. Several subjects are asked to pose for all of the seven actions multiple times. The collected depth images are used to train a Random Forest using selected parameters such as the depth of the tree, the number of features, the number of decision trees. At the recognition stage, the software outputs the recognized driving activity for each input depth image. Figure 1 provides a collection of typical depth images for the seven driving actions defined above. The gray level represents the depth value (the nearer to the camera, the smaller the gray level value).

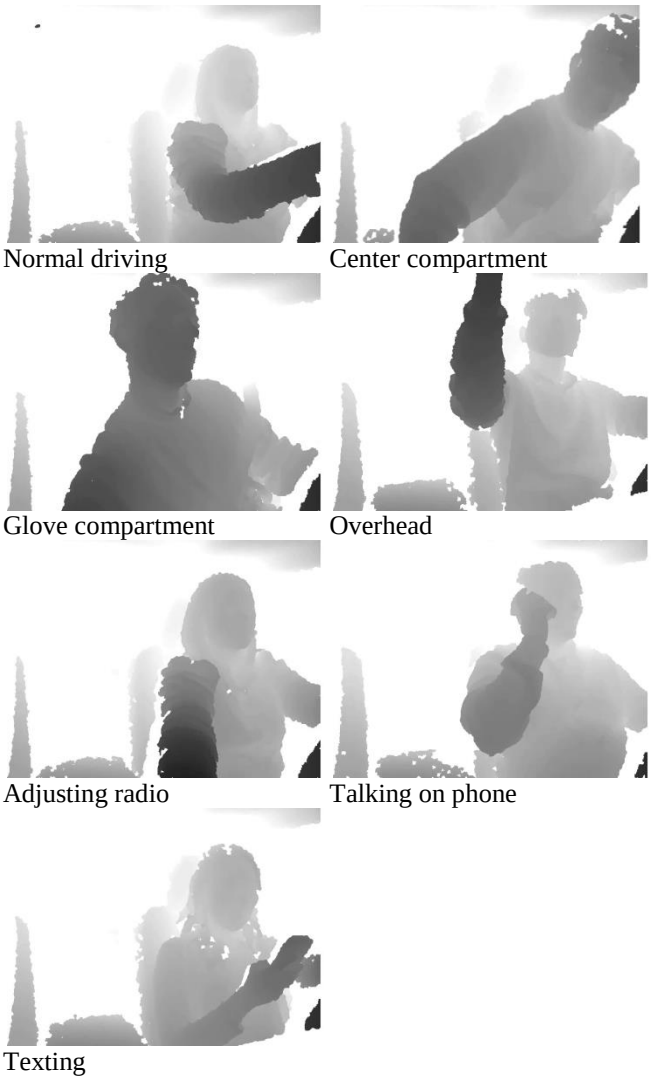


Figure 1. Driving activity categories

Data Cleaning

Before entering the training stage, collected images need to be cleaned up. The cleanup process includes three aspects. One is to delete low quality training images. The second aspect is to clean up the video so it only contains key poses of the driver. The third aspect relates to the limitation of the classifier. Classifying various activities by left-handed drivers as the same category as regular (right-handed) drivers is challenging to the classifier. Thus, this is not the focus of this work and we remove such images from our training and testing sets.

First, noisy depth images are removed. Improper system settings can produce low quality depth images. Due to the nature of the depth sensor, direct sun light can saturate the sensor and produce noisy depth images which need to be removed from the training image set. The third image in Figure 2 contains an example of noisy images. We can see that pixels corresponding to the torso and parts near the folded arm in the image are statured. The depth value is not smooth and contains large blobs of bright regions. The system set up may not have been in a good condition when collecting these data and are not representative situations for normal use cases. These images will be deleted from the training set.

The second aspect of the cleanup is to remove unrelated image frames, thereby keeping only representative key frames for the pose. This is important because the training process is based on individual images instead of the whole video; each training image affects the classifier's performance. Irrelevant images enclosed in the training video may confuse the training process and smear the boundary between different driving actions. A good example of irrelevant images are transit images at the beginning or ending of the videos where the user either getting ready or finishing up a particular activity. The first image in Figure 2 provides example images between stages. Both the first and the second images are taken from the same video sequence in "reaching for center compartment" pose. The first image represents a transit stage, where the driver is at an initial stage and getting ready for posing for "center compartment". This type of transit stage images are deleted from the training set. The second image is the valid training image for center compartment.

c

Figure 2 Training data cleaning. Fig 2a is a transit state image and Fig 2b is desired center compartment pose. Fig 2c is a noisy depth image, thus deleted.

Another source of training error comes from the fact that there exists a wide variety for how a certain activity can be executed. For example, some activities can be posed by either the left hand or the right hand and both cases make perfect sense to humans. Though the "mirrored" images represent the same syntax to human, it represents significantly different data from the classifier point of view. From our test, usually one of them is recognized as the correct activity while the other has a high chance to be classified as another activity category. Figure 3 provides such an example. Images fall this category will be deleted. For simplicity, we require right hand posing in the current paper, the right image in Figure 3 will not be used for our work.

Figure 3. Training error for the same activity posed by different hands

Besides posing with different hands, another possibility of such a case is driving activities that require a wide range of motion of body parts. Figure 4 contains such an example. The user is asked to pose for searching in the overhead compartment. In the same video, we can see the relative position between the arm and the head has changed. Though both poses are correct, we have to choose one of them as the training image and ignore the other. Here, we restrict that the overhead posing must be similar to the left image, and ignore frames resembling the right image in the training data.

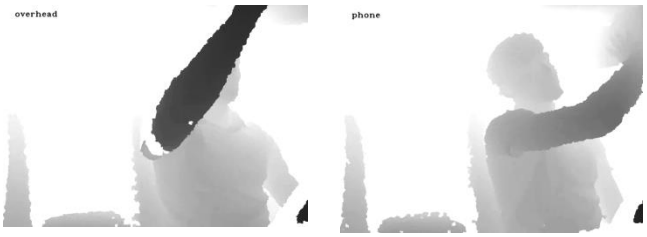


Figure 4. Training error for same activity and the same arm with wide moving range

RANDOM FOREST TRAINING

The Random Forest is an ensemble classification method. It takes output from a collection of weak classifiers (decision trees), and combines the result by voting. A decision tree is a binary tree which splits a node with the maximum information gain. The Random Forest is named as such, because the tests that split each node are selected randomly.

Node Split

In our training system, the decision tree is calculated by splitting each node with the test that maximizes the information gain [11]. The node split if the given function F(p) value exceeds a given threshold τ.

$$\mathrm{F(p)} = |F_1|^{-1} \sum_{q \in F1} I^f(q) - |F_2|^{-1} \sum_{q \in F2} I^f(q)$$

F_1 and F_2 are two randomly selected regions; they can be any rectangular shaped regions as long as the area is smaller than the defined maximum region size. We use 100 x 100 here. I(q) is the value of a feature channel for which we select the depth value at point q. Values of $|F_1|$ and $|F_2|$ are normalized to fit in range (0 ~ 1).

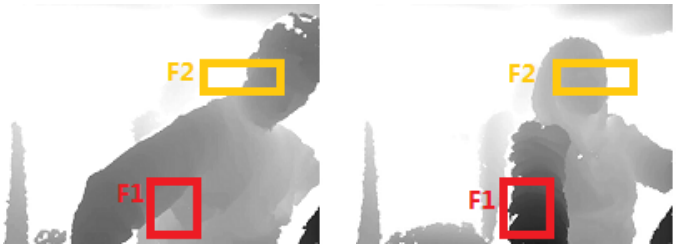


Figure 5 Calculation of Node Split Function F(x)

Information Gain

Each split defines an information gain value. Let N be the original tree node, L be the left child node and R be the right tree node. H(x) be the entropy value of a node.

The entropy for node X is

$$H(X) = -\sum_i p(i,X) \log p(i,X)$$

Where p(*i*, *X*) is the probability of category *i* in node *X*. The information gain for a split is:

$G(N) = H(N) – H(L) – H(R)$

Decision Tree Training

Each decision tree is trained using the split function and the information gain function.

Pick threshold τ

Do

{

Do

{

Pick F_1 and F_2

Calculate F(p)

Split N if $F(p) > \tau$

{

Calculate information gain G(N)

}

} Until (stop criteria 1 met)

Test = test that maximizes G(N)

} Until (stop criteria 2 met)

Stop criteria 1: the set is unanimous; or the number of samples in current set is smaller than the minimum; or the tree depth is larger than the maximum. In our training process, the minimum split set size is defined as 20; maximum tree depth is defined as 15.

Stop criteria 2: test number larger or equals maximum test number. The maximum is defined as 1000. The number of trees trained is 6.

RANDOM FOREST ERROR PATTERNS

The Random Forest is one of the most robust and precise classification algorithms. However, due to the complexity of the problem, there are still recognition errors. Based on the ground-truth data collected, several patterns of errors are observed. The sequences are time based.

Error Sequence Representation

Let SI represent the ground truth of an input depth image sequence, Sr represent an output depth image sequence, T represent the length of the sequence.

$SI(i) \neq Sr(i)$ are error cases, where i in [1, 2, …T].

To simplify the notation, let SI be an monotonous sequence, where all activities are As. The error cases are those in Sr (i), whose recognized activities are not As. For example:

Sr (i) = A A B B A A A A

represents an error pattern when activity A is recognized as B when i = 3, 4.

A more compact representation:

$S_r(i)$ = A B(2) A

where the number after an error case (in parenthesis) represents the length of the error sequence.

Error Patterns Observed

The following error patterns are observed from our testing.

P1: A----B(n)

P2: B----A(n)

P3: A----B(n)-----A

P4: A----B(n)-----C(m)----A

P5: A----B(n)-----C(m)----A----C(l)

where n, m, l are integral numbers. We will introduce how to reduce some of these error cases by a state based inference system.

STATE MACHINE ENHANCEMENT

In this section we introduce the state based inference tool, the state machine. The state machine is a post processing tool that reduces recognition error based on temporal analysis of an image sequence. Each state represents one of the seven driving activity categories. The image sequence feeds into the state machine and each image may change the status of the state machine. The state machine is then queried on the fly for the inference state.

The state machine inference focuses on the transition among different driving activities. Based on current state machine status and constraints, it decides whether to accept or reject the transition.

Transit Graph

Each node represents one state and the edges between states represent the transit probability among nodes. If there is no direct edge connecting two nodes, then the transit probability is zero, otherwise it is non-zero. The connectivity of the graph is formulated by physically possibility to complete such a transition. For example, it is impossible to transit from phone to radio without going to Normal first, therefore the transit probability between Phone and Radio is zero. And, both Phone and Radio have a non-zero transit probability to state Normal. Figure 6 is the transit graph we used. Transit between two nodes has a non-zero probability only when they belong to the same sub-graph enclosed by a red circle.

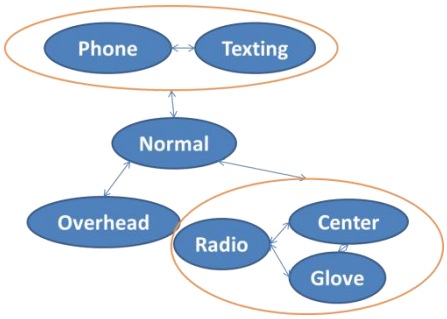


Figure 6 Transit Graph

Confidence Level

A variable *Confidence Level*, a floating point value in the range of [0, 1], is introduced to represent the confidence of the state being the true recognition result. A new state usually starts with the confidence of 1.0 and increase or decrease depends on the oncoming states. The value is capped to 1.0 and does not go under zero. The state inference result is the state with the largest confidence level; if more than one such state exists, we pick the one with the latest time stamp.

The state machine starts empty. Each oncoming driving activity modifies the state machine by updating the confidence values. If the corresponding state does not exist, it is added to the state machine with a positive confidence value. Otherwise, the confidence value of the state is increased (by 0.2 in our system). The confidence values of other states are decreased. If the confidence goes to zero, remove the state from the state machine. If the new state is different from the previous state, check the transit graph. If the transit is of high cost, the initial state is set to zero. Otherwise the value is set to 1.

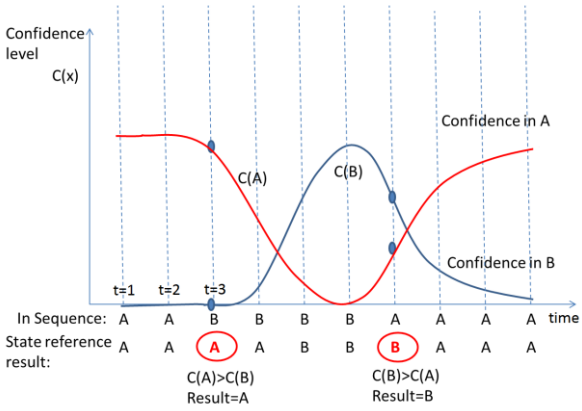


Figure 7. Confidence Level

Figure 7 represents a state machine with two states A and B and the time sequence of their confidence values. The horizontal axis is time and vertical axis is confidence level. As we can saw the recognized sequence starts with A, with confidence value of 1. Then at t = 3, the first state transit from A to B happens. Because this is a high cost transit after referring to the transit graph, the new state B starts from a low confidence. The confidence of A starts to decrease. At t = 4, another B is observed and same process is repeated. At this time, though confidence of A decreases and confidence of B increases. Since the former is still larger than the latter, the state inference value remains as A. This changes at t = 4 where the confidence value of B exceeds that of A, and the output flipped and becomes B. At t = 7, we see another state transit, this time the transit from B to A. Confidence of B decreases and confidence of A increases. Finally, at t = 8, confidence value of A exceeds that of B and the output flips again from B to A.

The initial confidence value for a new state is set to zero if it belongs to a high cost transit. This way, we need to see more frames of such state before gaining enough confidence to transit to the new high cost state. Accordingly, the initial confidence value for a new state is set to one if it is a low cost transit. The transit to the new state is immediately approved with no latency as penalty.

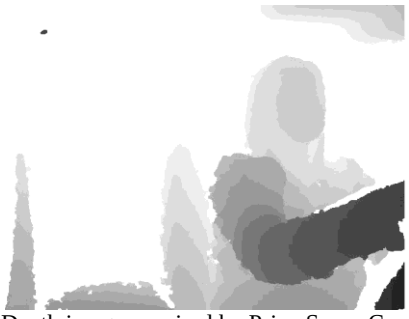
TEST RESULTS

We now show performance of our method. Two depth sensors were used for our experiment: PrimeSense Carmine 1.09 and SoftKinetic DepthSense 325. The camera parameters for the two are listed in Table 1. We develop the framework based on the open project Point Cloud Library (or PCL).

Table 1. Camera Parameters

	SoftKinetic DepthSense 325	**PrimeSense Carmine 1.09**
Range	0.15m ~ 1m	0.35m ~ 1.4m
Field of View	74º x 58º	57.5º x 45º
Depth Image Resolution	320 x 240	640 x 480
FPS	18 ~ 21	7 ~ 8

Depth image acquired by SoftKinetic DepthSense 325 sensor.



Depth image acquired by PrimeSense Carmine 1.09

Figure 8. Depth images acquired by different depth sensors. The PrimeSense Carmine produces higher quality images with larger size compared to SoftKinetic DepthSense, at the cost of lower FPS.

The computer used for training and testing is Intel(R) Core™i7-3630QMCPU@2.4GHz, with 8GB memory. We have 15 participants, approximately 6000 seconds 49K frames of video.

Table 2 compares the overall performance of the two depth sensors: PrimeSense and Kinect. We also study the effect of the number of features per node (1K features per node vs 10K features per node). Much more training time is required for the latter. We can see the PrimeSense camera outperforms the SoftKinetic camera in both cases. Therefore we select the PrimeSense camera as our test tool and the following tests are carried out by using the PrimeSense camera. Also, we observed from Table 1 that increasing the number of features does not improve the result significantly for the PrimeSense sensor. Thus, we use 1K features per node as our selection to reduce training time without compromising the result.

Table 4. Confusion Matrix for Random Forest

	C	G	N	O	P	R	T
C	-	-	-	-	-	-	-
G	-	-	-	-	-	-	-
N	-	-	-	-	3.04	-	1.05
O	-	-	-	-	5.3	-	-
P	-	-	7.4	-	-	-	1.35
R	-	-	1.53	-	-	-	-
T	-	-	0.1	-	2.0	-	-

Table 2. Comparison of Recognition Rate between Different Depth Sensor

Camera	Featuers/Node 1K (%)	Features/Node 10K(%)
PrimeSense	87.06	88.00
SoftKinetic	77.25	82.75

Table 3 compares the recognition rates with and without state machine enhancement for different activities. The value is the success rate, the higher the value the more accurate the result. We can see in most of the cases state inference outperforms using the Random Forest only, except the texting activity. This explains one limitation of the state inference algorithm. From Table 4 we can see texting can be confused as normal driving activity. If the confusion happens at the beginning of the video, it takes the state inference algorithm some time before it recovers from the error. This recovery period is a side effect of state inference when the starting frames are recognized incorrectly; the longer the time, the worse the result. If the activity is mistaken as an activity from a different activity group, the error is larger because we penalize such transit and it takes even longer time to recover.

Table 3. Comparison between the Random Forest and the State Inference Enhancement

Activity	Random Forest (%)	State Inference(%)
Center	94.59	96.2
Glove	98.05	98.26
Normal	94.8	97.8
Overhead	93.12	93.6
Phone	89.6	89.8
Radio	97.5	98.27
Texting	97.5	95

Table 4 and Table 5 display the confusion matrices for Random Forest and Random Forest with State Reference Enhancements. To save place, we use initials to represent each driving activity name. The number is the percentage of the confusion value; it is calculated by dividing the number of error cases by the total frames. The lower the value the more accurate the result is. Zero percent is marked as- to avoid cluttering the table. The left-most errors lists the actual activities; the first row lists the identified activities. As discussed previously, if an error happens from the beginning of the activity, it would be costly to recover. The case where state inference has a higher confusion rate belongs to such cases. Fortunately, these cases occur less frequently than other cases as pointed in Section 5.2. Usually P3, P4, and P5 (as defined in Section 5.2) cause he major **(%)** trouble cases.

Table 5. Confusion Matrix for State Reference (%)

	C	G	N	O	P	R	T
C	-	-	-	-	-	-	-
G	-	-	-	-	-	-	-
N	-	-	-	-	1.15	-	0.6
O	-	-	-	-	5.29	-	-
P	-	-	7.87	-	-	-	1.15
R	-	-	1.29	-	-	-	-
T	-	-	3.2	-	1.77	-	-

CONCLUSION AND FUTURE WORKS

In this paper, we introduce an online real-time driver activity recognition system. The Random Forest is employed in both training and recognition. To further reduce recognition errors we propose a state machine as a post process procedure. Transit from one driving activity to another is either accepted or rejected based on the history so far and the possibility of the transit itself. We discuss the Random Forest parameter selection, updates of confidence level, state transit graph, and fine tunes of state machine parameters. Finally we have presented and analyzed the test results and discuss the advantage and limitation of the state machine.

There are several improvements that can be applied to the framework. It is important to improve the Random Forest training to accommodate the left hand pose and right hand pose dilemma. One solution would be detect the position of head and divide the training set based on hand and head relative position. The training is carried out separately on the two. Another future research direction is adding new feature channels such as face normal direction. This can help distinguish current easy confused driving activities. For example, texting is easy to get confused with talking on the phone. But because their face orientations are quite different, it will be easier to separate them after adding normal direction feature.

ACKNOWLEDGMENTS

We thank Trevor Sarratt for initial implementation of the decision forest algorithm. We also want to thank the volunteers' time to help to prepare the training datasets.

REFERENCES

1.Healey, J., Wang, C., Dopfer, A., and Yu, C., 2012. M2M gossip: why might we want cars to talk about us?. *Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (AutomotiveUI '12). ACM, pp. 265-268.

2.Baker, S., Matthews, I., Xiao, J., Gross, R., Kanade, T. and Ishikawa, T. 2004. Real-time non-rigid driver head tracking or driver mental state estimation, *11th World Congress on Intelligent Transportation Systems.*

3.Ji, Q., Yang, X., 2002. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*. Volume 8 Issue 5, pp. 357-377

4.Doshi, A., Trivedi, M., M., 2009. On the roles of Eye Gaze and Head Dynamics in Predicting Driver's Intent to Change Lanes. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, No. 3

5.Zhu, Y., Fujimura, K., 2004. Head pose estimation for driver monitoring. *IEEE Intelligent Vehicles Symposium*, pp. 501 - 506

6.Murphy-Chutorian, E., Trivedi, M., M., 2009. Head Pose Estimation in Computer Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 31 Issue 4, pp. 607-626

7.Veeraraghavan, H., Bird, N., Atev, S. and Papanikolopoulos, N. 2007. Classifiers for driver activity monitoring. *Transportation Research. Part C*, vol. 15, no. 1, pp. 51-67.

8.Breiman, L., 2001. Random forests. *Machine Learning*. Vol 45, Issue 1, pp.5-32.

9.Criminisi, A., Shotton, J., Robertson, D., and Konukoglu, E., 2010.Regression forests for efficient anatomy detection and localization in ct studies. *Medical Computer Vision Workshop*

10.Gall, J., Yao, A., Razavi, N., Van Gool, L., and Lempitsky, V., 2011. tracking, and action recognition Hough forests for object detection. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 33, issue 11

11.Huang, C., Ding, X., and Fang, C., 2010. Head pose estimation based on random forests for multiclass classification. *ICPR*.

12.Fanelli, G., Gall, J., and Van Gool, L., 2011. Real Time Head Pose Estimation with Random Regression Forests *Computer Vision and Pattern Recognition*, pp. 617-624.

13.Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., and Blake, A., 2013. Efficient Human Pose Estimation from Single Depth Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2821-2840.

14.Shakhnarovich, G., Viola, P., and Darrell. T., 2003. Fast Pose Estimation with Parameter-Sensitive Hashing. *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2, pp. 750

15.Agarwal, A., Triggs, B., 2006. Recovering 3D Human Pose from Monocular Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 1, pp. 44-58